GrapheneOS

The Definitive Guide to Mobile Security

Take control. Protect your data. Own your privacy.

Privacy by Design | Strong Security | No Tracking No Surveillance | Open Source Transparent

ROKIBUL ISLAM

The Digital Panopticon

GrapheneOS: মোবাইল সিকিউরিটির পূর্ণাঙ্গ গাইড

লিখেছেন **রকিবুল ইসলাম**

প্রকাশকাল: মে ২০২৬

Edition: i-2026

© ২০২৬ রকিবুল ইসলাম। সর্বস্বত্ব সংরক্ষিত।

ঘোষণাপত্র: এই বইটি একটি স্বাধীন শিক্ষামূলক সংস্থান। লেখকের সাথে GrapheneOS ফাউন্ডেশনের সরাসরি কোনো সম্পর্ক নেই। সমস্ত ট্রেডমার্ক, লোগো এবং হার্ডওয়্যার রেফারেন্স তাদের নিজ নিজ মালিকদের সম্পত্তি। সাইবার নিরাপত্তার ক্ষেত্র দ্রুত পরিবর্তিত হয়; পাঠকদের মিশন-ক্রিটিক্যাল ডিপ্লয়মেন্টের জন্য অফিসিয়াল GrapheneOS ডকুমেন্টেশন পড়ার পরামর্শ দেওয়া হচ্ছে।

উৎসর্গ

এই বইটি উৎসর্গ শুধু সেইসব বিরল মস্তিষ্কের প্রতি,
যারা আজও মেটা-গুগলের নির্মিত বিভ্রমের সীমানার বাইরেও চিন্তা করতে জানে।

আইনি সতর্কবার্তা ও দায়মুক্তি (Legal Disclaimer)

শিক্ষামূলক উদ্দেশ্য: এই বইয়ে উপস্থাপিত যাবতীয় তথ্য, টুলস এবং পদ্ধতিসমূহ শুধুমাত্র ব্যক্তিগত গোপনীয়তা রক্ষা (Privacy Protection), সাইবার নিরাপত্তা নিয়ে পড়াশোনা এবং এথিক্যাল (Ethical) গবেষণার উদ্দেশ্যে প্রদান করা হয়েছে। কোনো বেআইনি কাজ বা রাষ্ট্রবিরোধী কার্যকলাপে এই তথ্যের ব্যবহারকে লেখক বিন্দুমাত্র উৎসাহিত বা সমর্থন করেন না।

পরিভাষার স্পষ্টীকরণ: বইয়ে ব্যবহৃত বিভিন্ন শব্দ—যেমন "Evasion", "Anonymity", "Compartmentalization", "Phantom Comms" বা "Threat Actor"—এগুলো গ্লোবাল সাইবার সিকিউরিটি ইন্ডাস্ট্রির স্ট্যান্ডার্ড টার্ম। এগুলো মূলত হ্যাকার, ক্ষতিকর ম্যালওয়্যার, সাইবার ক্রিমিনাল এবং কর্পোরেট ডেটা-ব্রোকারদের (যেমন মেটা বা গুগল) হাত থেকে সাধারণ ব্যবহারকারীকে বাঁচানোর ডিফেন্সিভ (প্রতিরক্ষামূলক) কৌশল বোঝাতে ব্যবহৃত হয়েছে; কোনো বৈধ রাষ্ট্রীয় তদন্ত বা আইনপ্রয়োগকারী সংস্থাকে ফাঁকি দেওয়ার উদ্দেশ্যে নয়।

সম্পূর্ণ দায়মুক্তি (No Liability): কোনো পাঠক যদি এই বইয়ের তথ্যের অপব্যবহার করে কোনো ব্যক্তি, প্রতিষ্ঠান বা রাষ্ট্রের ক্ষতিসাধন করেন অথবা কোনো দেশের আইন ভঙ্গ করেন, তবে তার সম্পূর্ণ আইনি ও আর্থিক দায়ভার (Liability) পাঠকের নিজের। লেখক বা এই প্রজেক্টের সাথে যুক্ত কেউ এর জন্য প্রত্যক্ষ বা পরোক্ষভাবে দায়বদ্ধ থাকবেন না।

স্থানীয় আইনের প্রতি শ্রদ্ধাশীলতা: এনক্রিপশন, ভিপিএন (VPN) বা কাস্টম অপারেটিং সিস্টেম ব্যবহারের ক্ষেত্রে নিজ নিজ দেশের (Jurisdiction) স্থানীয় আইন মেনে চলার সম্পূর্ণ দায়িত্ব পাঠকের।

GrapheneOS: মোবাইল সিকিউরিটির পূর্ণাঙ্গ গাইড

মূল সূচিপত্র

ভূমিকা

ডিজিটাল প্যানোপটিকন ১০

GrapheneOS: একটি প্যারাডাইম শিফট

দ্য ব্লুপ্রিন্ট অব রেসিস্ট্যান্স: এই বইটি কীভাবে ব্যবহার করবেন

পর্ব ১: ফাউন্ডেশন

অধ্যায় ১: ভূমিকা

১.০ ফাউন্ডেশনে স্বাগতম ১৬

১.১ GrapheneOS কী?

আর্কিটেকচারের উৎপত্তি

De-Googling বনাম Hardening: গুরুত্বপূর্ণ পার্থক্য

অধ্যায় ১: সেকশনসমূহ

১.২ হার্ডওয়্যারের প্রয়োজনীয়তা এবং পিক্সেলের মনোপলি ২২

চূড়ান্ত পরিহাস: সিলিকন থেকে সফটওয়্যারকে আলাদা করা

ঐতিহাসিকভাবে কেন শুধু পিক্সেল?

সাপোর্টেড পিক্সেল ডিভাইসের তালিকা

২০২৬ সালের মাইলফলক: মনোপলি ভাঙা

১.৩ প্রি-ইন্সটলেশন এবং OEM আনলকিং ২৭

পূর্বশর্ত: প্রয়োজনীয় উপকরণ সংগ্রহ

ক্যারিয়ার-লক সতর্কতা

ধাপে ধাপে: OEM আনলকিং এনাবল করা

১.৪ ওয়েব ইন্সটলার পদ্ধতি ৩২

ব্রাউজার কম্প্যাটিবিলিটি

পর্যায় ১-৫: Fastboot → Flash → Lock Bootloader → First Boot

১.৫ কমান্ড লাইন ইন্টারফেস (CLI) ইন্সটলেশন ৩৮

ধাপ ১-৭: Dependencies → Fastboot → Verify → Flash → Lock

১.৬ পোস্ট-ইন্সটলেশন: Verified Boot এবং Auditor App ৪৬

হলুদ সতর্কবার্তা স্ক্রিন

প্রাথমিক সেটআপ এবং OEM আনলকিং নিষ্ক্রিয় করা

Auditor App: হার্ডওয়্যার-ব্যাকড অ্যাটাস্টেশন

১.৭ ফার্স্ট বুট: ইউজার ইন্টারফেস এবং সেটিংস নেভিগেশন ৫১

জেসচার নেভিগেশন

ব্যাকগ্রাউন্ড রেডিও ট্রান্সমিশন সুরক্ষিত করা

অথেনটিকেশন হার্ডেনিং: পাসওয়ার্ড এবং পিন স্ক্র্যাম্বলিং

১.৮ ডিফল্ট GrapheneOS অ্যাপসমূহ ৫৭

Vanadium: দ্য হার্ডেনড ব্রাউজার

GrapheneOS Camera: টেলিমেট্রির চেয়ে প্রাইভেসির অগ্রাধিকার

পিডিএফ ভিউয়ার এবং অ্যাপস রিপোজিটরি

১.৯ দ্য মাইন্ডসেট: বেসিক প্রাইভেসি অভ্যাস (OpSec 101) ৬২

সার্চ ইঞ্জিন, যোগাযোগ, ইমেইল, পারমিশন, অতিরিক্ত সোশ্যাল শেয়ারিং

পর্ব ২: ফোর্ট্রেস

অধ্যায় ২: ভূমিকা

২.০ অ্যাডভান্সড OpSec মাইন্ডসেট ৬৯

আপগ্রেডেড থ্রেট মডেল

মূল দর্শন: অ্যাটাক সারফেস রিডাকশন

অপারেটরের শৃঙ্খলা

অধ্যায় ২: সেকশনসমূহ

২.১ আপস না করেই De-Googling: Sandboxed Google Play ৭৪

২.২ গুগলকে ধোঁকা দেওয়া: লোকেশন এবং নেটওয়ার্ক রাউটিং ৮০

২.৩ Storage Scopes: ফেইক স্টোরেজ দিয়ে অ্যাপগুলোকে বোকা বানানো ৮৪

২.৪ Contact Scopes: সোশ্যাল অ্যাপগুলোর Sandboxing ৮৯

২.৫ দ্য আল্টিমেট কিল সুইচ: Per-App নেটওয়ার্ক টগলস ৯৪

২.৬ এক্সট্রিম কম্পার্টমেন্টালাইজেশন: ইউজার প্রোফাইল ৯৮

২.৭ ফাইন্যান্সিয়াল সিকিউরিটি এবং Play Integrity ১০৫

২.৮ ফিশিংয়ের বিরুদ্ধে প্রতিরোধ: App Link Verification ১১০

২.৯ ফিজিক্যাল আর্মার: হার্ডওয়্যার সিকিউরিটি কিস ১১৪

২.১০ নেটওয়ার্ক অ্যানোনিমিটি: ওয়াই-ফাই ট্র্যাকিং পরাস্ত করা ১১৯

২.১১ এনক্রিপ্টেড টানেল: VPN এবং কিল সুইচ ১২৫

২.১২ প্রাইভেট DNS: ফোন বুক এনক্রিপ্ট করা ১৩০

২.১৩ বেসব্যান্ড ওয়ারফেয়ার: LTE-Only মোড এবং Stingrays ১৩৪

২.১৪ ফ্যান্টম কমিউনিকেশন: ভার্চুয়াল eSIM এবং VoIP ১৩৯

২.১৫ ট্রাভেল সিকিউরিটি: BFU বনাম AFU বোঝা ১৪৩

২.১৬ দ্য ডেডম্যান্স সুইচ: অটো-রিবুট টাইমার ১৪৮

২.১৭ দ্য নিউক্লিয়ার অপশন: Duress PIN ১৫২

২.১৮ লকিং দ্য গেটস: USB-C পোর্ট কন্ট্রোল ১৫৬

পর্ব ২ উপসংহার: দ্য ফোর্ট্রেস ব্লুপ্রিন্ট ১৬১

পর্ব ৩: ব্যাটলফিল্ড

অধ্যায় ৩: ভূমিকা

৩.০ রুলস অব এনগেজমেন্ট ১৬৫

রেড টিম বনাম ব্লু টিম ফ্রেমওয়ার্ক

অ্যানালাইজিং দ্য কিল চেইন

অধ্যায় ৩: সেকশনসমূহ

৩.১ অ্যাটাক সারফেস সম্পর্কে বোঝা ১৬৯

৩.২ দৃশ্যপট ১: Physical Forensic Extraction ১৭৪

৩.৩ দৃশ্যপট ২: Cold Boot RAM Extraction ১৮২

৩.৪ দৃশ্যপট ৩: Zero-Click Warfare (Pegasus) ১৮৮

৩.৫ ডিপ ডাইভ: ARM Memory Tagging Extension (MTE) ১৯৬

৩.৬ ডিপ ডাইভ: hardened_malloc অ্যালোকেটর ২০৫

৩.৭ দৃশ্যপট ৪: MitM এবং Baseband ইন্টারসেপশন ২১০

৩.৮ দৃশ্যপট ৫: সাপ্লাই চেইন অ্যাটাক ২১৬

৩.৯ ডিপ ডাইভ: Exec Spawning বনাম Zygote ২২২

৩.১০ দৃশ্যপট ৬: ম্যালিশিয়াস অ্যাপ এস্কেপ ২২৯

৩.১১ ডিপ ডাইভ: IOMMU এবং হার্ডওয়্যার আইসোলেশন ২৩৬

পর্ব ৩ উপসংহার: দ্য আনব্রেকএবল ফোর্ট্রেস ২৪১

অ্যাপেন্ডিক্স (পরিশিষ্ট)

অ্যাপেন্ডিক্স A: ডেভেলপারদের ডোমেইন **২৪৫**

A.1 GrapheneOS কম্পাইল করা: পূর্বশর্তসমূহ **২৪৮**

A.2 GrapheneOS কম্পাইল করা: বিল্ড প্রসেস **২৫২**

অ্যাপেন্ডিক্স B: এন্টারপ্রাইজ ডিপ্লয়মেন্ট এবং MDM **২৬২**

অ্যাপেন্ডিক্স C: সার্ভার আর্কিটেকচার এবং CAKE **২৬৮**

অ্যাপেন্ডিক্স D: মাস্টার রেফারেন্স বিবলিওগ্রাফি **২৭৩**

উপসংহার

প্রাইভেসির ভবিষ্যৎ **২৯২**

প্রস্তাবনা: ডিজিটাল প্যানোপটিকন

রকিবুল ইসলাম রনি কর্তৃক

অফেন্সিভ সিকিউরিটি এবং রেড টিম অপারেশনে বিশেষজ্ঞ একজন পেনিট্রেশন টেস্টার।

rokibulroni.com | mail@rokibulroni.com

আপনার হাতে এমন একটি ডিভাইস রয়েছে, যা আপনার সম্পর্কে আপনার নিজের চেয়েও বেশি তথ্য জানে।

আধুনিক যুগে, "স্মার্টফোন" ধারণাটি একটি মারাত্মক সেকেলে ভুল নাম। আপনার পকেটে থাকা কাঁচের আয়তক্ষেত্রটি আসলে কোনো ফোন নয়; এটি একটি সর্বক্ষণ সচল, মাল্টি-সেন্সর টেলিমেট্রি বিকন। এটি একটি অত্যন্ত উন্নত ইন্টেলিজেন্স-গ্যাদারিং টার্মিনাল, যা আপনি স্বেচ্ছায় নিজের জীবনের সবচেয়ে ব্যক্তিগত স্থানগুলোতে জায়গা দিয়েছেন। এটি চালু করলেই আপনার লোকেশন ম্যাপ হতে শুরু করে, আপনার সোশ্যাল গ্রাফ ক্যাটালগ হয়, বায়োমেট্রিক্স আর্কাইভ হয় এবং আপনার ব্যক্তিগত আচরণকে পণ্যে রূপান্তর করা হয়। আপনি অজান্তেই ডিজিটাল প্যানোপটিকনে প্রবেশ করেছেন—এটি সার্ভিল্যান্স ক্যাপিটালিজমের এমন এক সর্বব্যাপী, অদৃশ্য স্থাপত্য যেখানে আপনার মেটাডেটাই হলো প্রধান মুদ্রা, এবং আপনার সম্পূর্ণ প্রাইভেসি হলো এমন একটি বাধা, যাকে সুপরিকল্পিতভাবে নির্মূল করা হয়।

আধুনিক মোবাইল কম্পিউটিংয়ের ভিত্তিগত মনোপলি—স্ট্যান্ডার্ড Android এবং Apple-এর iOS—ঠিক সেভাবেই কাজ করে যেভাবে এদের ডিজাইন করা হয়েছে। "এনক্রিপশন" এবং "প্রাইভেসি"-এর মতো স্বস্তিদায়ক শব্দের আড়ালে থাকা মিলিয়ন ডলারের মার্কেটিং ক্যাম্পেইন সত্ত্বেও, এই কমার্শিয়াল অপারেটিং সিস্টেমগুলোর রন্ধ্রে রন্ধ্রে আপস করা রয়েছে। এগুলো মূলত কর্পোরেট ডেটা হারভেস্টিংয়ের ওপর ভিত্তি করে তৈরি, যা এমন সব প্রোপাইটারি সার্ভিস দ্বারা নিয়ন্ত্রিত হয় যাদের পুরোপুরি অডিট করা অসম্ভব। একইসাথে এগুলো টেলিকমিউনিকেশন কর্পোরেশনগুলোর বেঁধে দেওয়া বেসব্যান্ড আর্কিটেকচারের সাথে ওতপ্রোতভাবে যুক্ত। এগুলো আপনাকে নিরাপত্তার একটি ভ্রান্ত ধারণা দেয়; ইউজার-ইন্টারফেসে এমন কিছু টগল (toggle) অপশন দেয় যা কার্নেল-লেভেলের অন্তর্নিহিত টেলিমেট্রি বন্ধ করতে খুব একটা কাজেই আসে না।

যখন অত্যাধুনিক থ্রেট অ্যাক্টর, জটিল Zero-day exploit চেইন, বা রাষ্ট্র-পরিচালিত নজরদারি কাঠামোর সম্মুখীন হতে হয়, তখন এই স্ট্যান্ডার্ড কমার্শিয়াল মোবাইল প্ল্যাটফর্মগুলো ডিফল্ট অবস্থায় পর্যাপ্ত প্রতিরোধ গড়তে প্রায়শই ব্যর্থ হয়। যদিও শক্তিশালী বিকল্প রয়েছে—যেমন লকডাউন মোডে কনফিগার করা Apple-এর iOS, Google Pixel-এ থাকা অত্যন্ত হার্ডেনড (hardened) স্টক Android, অথবা ডেডিকেটেড Linux মোবাইল এনভায়রনমেন্ট—তবে প্রকৃত নিরাপত্তার কাছাকাছি পৌঁছানোর জন্য এগুলোকে অত্যন্ত সচেতনভাবে এবং সুপরিকল্পিত উপায়ে কনফিগার করতে হয়।

এটিই সমসাময়িক ডিজিটাল যুদ্ধক্ষেত্রের রূঢ় বাস্তবতা। তবে এটি এমন কোনো বাস্তবতা নয় যা আপনাকে বাধ্য হয়ে মেনে নিতে হবে।

কেন আমি এই বইটি লিখেছি

অফেন্সিভ সিকিউরিটি এবং রেড টিম অপারেশনে বিশেষজ্ঞ একজন পেনিট্রেশন টেস্টার হিসেবে আমার প্রাত্যহিক কাজ হলো নেটওয়ার্কে অনুপ্রবেশ করা, সিকিউরিটি প্রোটোকল বাইপাস করা এবং সংবেদনশীল তথ্য বের করে আনা। টার্গেটেড অ্যাটাকের মুখে কমার্শিয়াল অপারেটিং সিস্টেমগুলো কত সহজেই মুখ থুবড়ে পড়ে, তা আমি নিজের অভিজ্ঞতা থেকে দেখেছি। কমার্শিয়াল স্মার্টফোন ইন্ডাস্ট্রি আপনার নিরাপত্তার সাথে কাঠামোগতভাবেই সাংঘর্ষিক; এটি প্রকৃত ও যাচাইযোগ্য সিকিউরিটির চেয়ে বাধাহীন সুবিধা এবং ডেটা এক্সট্রাকশনকেই বেশি অগ্রাধিকার দেয়।

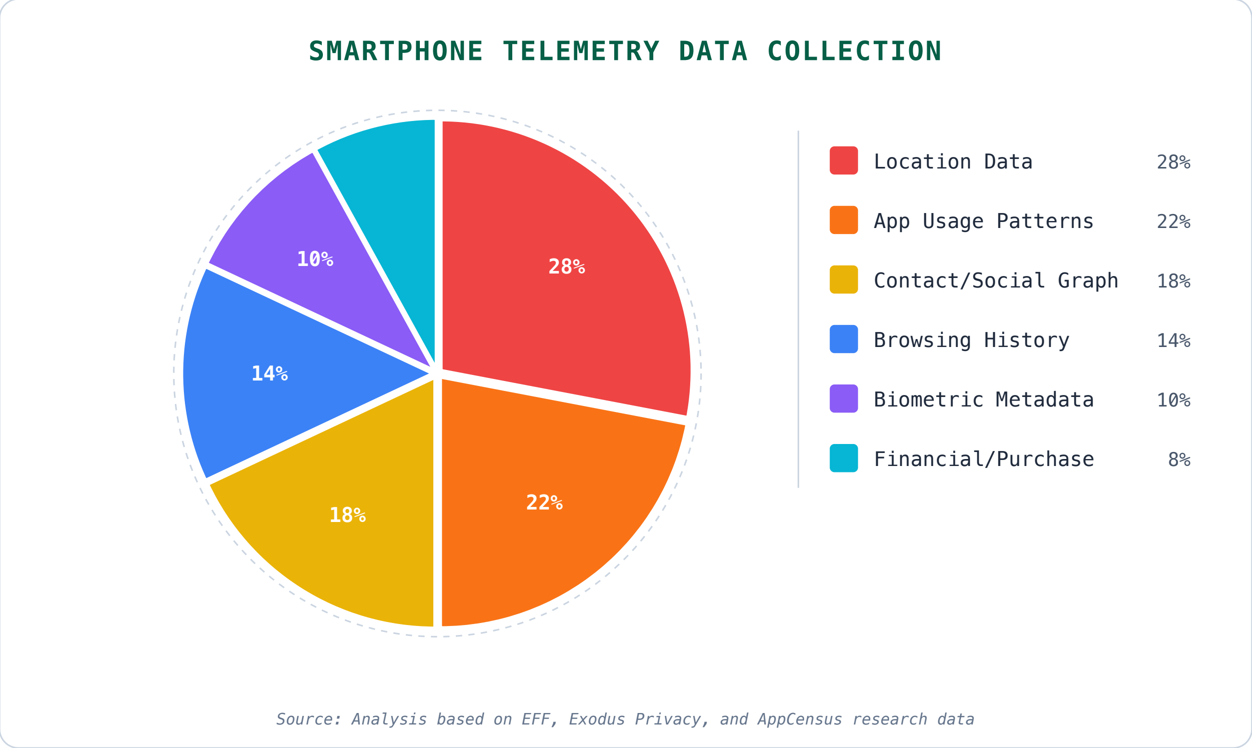
আমি এই বইটি লিখেছি কারণ সাধারণ ব্যবহারকারীরা—এমনকি অনেক আইটি পেশাদাররাও—অজান্তেই তাদের মোবাইল ডিভাইসগুলোকে Forensic extraction, Zero-click exploit এবং নজরদারি পুঁজিবাদের সামনে কতটা অরক্ষিত করে রেখেছেন, তা সম্পর্কে তারা সম্পূর্ণ অসচেতন। এই গাইডটি কোনো তাত্ত্বিক একাডেমিক বই নয়; এটি একটি ট্যাকটিক্যাল ব্লুপ্রিন্ট। আমি আমার অফেন্সিভ সিকিউরিটির অভিজ্ঞতাকে একটি ডিফেন্সিভ ডকট্রিনে রূপান্তরিত করেছি, এবং GrapheneOS ব্যবহার করে এমন একটি মোবাইল এনভায়রনমেন্ট তৈরি করেছি যা আমি ও আমার সহকর্মীরা ফিল্ডে যে মাত্রার অস্ত্র (cyber weapons) ব্যবহার করি, ঠিক সেই মাত্রার আক্রমণ প্রতিহত করতে সক্ষম। এই বইটির মূল উদ্দেশ্য হলো ডিফেন্ডার বা সুরক্ষাকারীর হাতে আর্কিটেকচারাল সুবিধা ফিরিয়ে দেওয়া।

GrapheneOS: একটি প্যারাডাইম শিফট

বছরের পর বছর ধরে, সাইবার সিকিউরিটি ইন্ডাস্ট্রি এই গভীর ক্ষতে ভাসা-ভাসা ব্যান্ডেজ লাগানোর চেষ্টা করে এসেছে—কমার্সিয়াল VPN, হিউরিস্টিক অ্যান্টিভাইরাস স্ক্যানার এবং ভিত্তিমূল থেকেই পচে যাওয়া অপারেটিং সিস্টেমের ওপর তৈরি করা এনক্রিপ্টেড মেসেজিং অ্যাপ্লিকেশনের মাধ্যমে। এগুলো আসলে কোনো পূর্ণাঙ্গ সমাধান নয়। একটি দুর্বল ভিত্তির ওপর আপনি কখনোই সুরক্ষিত দুর্গ গড়তে পারবেন না।

GrapheneOS কোনো সাধারণ "Custom ROM" নয়। এটি কোনো শখের প্রজেক্ট নয় যা আইকন প্যাক বদলাতে বা ক্যারিয়ার রেস্ট্রিকশন বাইপাস করতে ডিজাইন করা হয়েছে। GrapheneOS হলো মোবাইল সিকিউরিটি আর্কিটেকচারে একটি বৈপ্লবিক ও গাণিতিকভাবে নিখুঁত প্যারাডাইম শিফট। সিকিউরিটি রিসার্চার ড্যানিয়েল মিকের ধারণায় তৈরি এই সিস্টেমটি হলো

Android Open Source Project (AOSP)-এর একটি সম্পূর্ণ ও আপসহীন পুনর্গঠন, যাকে বেয়ার মেটাল (bare metal) থেকে এমনভাবে ইঞ্জিনিয়ারিং করা হয়েছে যেন এটি আধুনিক টেলিকমিউনিকেশনের সর্বব্যাপী ট্র্যাকিং অবকাঠামো থেকে সম্পূর্ণ স্বাধীনভাবে কাজ করতে পারে।



কঠোরভাবে "Privacy by Default" নীতিমালার অধীনে পরিচালিত হয়ে, GrapheneOS একটি জিরো-টেলিমেট্রি এনভায়রনমেন্ট নিশ্চিত করে। এটি কাঠামোগতভাবেই ক্ষতিকারক ট্র্যাকিং সার্ভিসগুলোর ইন্টিগ্রেশন প্রত্যাখ্যান করে এবং সাইড-চ্যানেল বায়োমেট্রিক নজরদারিকে অকার্যকর করে দেয়। তবে সত্যিকারের সুরক্ষার জন্য অপ্রতিরোধ্য ডিফেন্স প্রয়োজন। GrapheneOS এমন একটি কৌশল তৈরি করেছে যা স্পষ্টভাবে অজানা হুমকির বিরুদ্ধে সুরক্ষার জন্যই ডিজাইন করা হয়েছে। এর কাস্টম হার্ডেনড মেমরি অ্যালোকেটর (`hardened_malloc`), কঠোর অ্যাটাক সারফেস রিডাকশন এবং শক্তিশালী Sandboxing-এর কারণে এটি নিশ্চিত করে যে, কোনো আক্রমণকারী যদি কোনো দুর্বলতা খুঁজেও পায়, তবুও একটি নির্ভরযোগ্য এক্সপ্লয়েট লেখা তাদের জন্য ভীষণ কঠিন হয়ে ওঠে।

তবে এটি মনে রাখা অত্যন্ত গুরুত্বপূর্ণ যে, Zero-day vulnerability সবসময়ই থাকবে এবং পর্যাপ্ত অর্থায়নে পুষ্ট Pegasus-লেভেলের থ্রেট অ্যাক্টররা সর্বদা একটি ভয়ঙ্কর হুমকি হয়েই থাকবে। GrapheneOS কোনো জাদুকরী ঢাল নয়, বরং এটি ব্যবহারকারীর অনুকূলে পরিস্থিতি ফিরিয়ে আনতে সক্ষম এমন সেরা ডিফেন্সিভ টুলগুলোর মধ্যে অন্যতম।

ঐতিহাসিকভাবে, এই স্তরের নিরাপত্তার জন্য হার্ডওয়্যারের ক্ষেত্রে কিছুটা আপস করার প্রয়োজন ছিল। GrapheneOS একচেটিয়াভাবে Google Pixel হার্ডওয়্যারের ওপর নির্ভরশীল ছিল—কারণ এটিই একমাত্র কমার্শিয়াল সিলিকন যা এর ক্রিপ্টোগ্রাফিক ভেরিফাইড বুট এবং সিকিউর এনক্লেভ সমর্থন করার মতো শক্তিশালী। কিন্তু দৃশ্যপট এখন বদলে গেছে। GrapheneOS ফাউন্ডেশন এবং Motorola Mobility-এর মধ্যকার ঐতিহাসিক ২০২৬ সালের অংশীদারত্ব একটি নতুন যুগের সূচনা করেছে। আমরা এখন আর কমার্শিয়াল ডিভাইসগুলোকে রেট্রোফিট করছি না; বরং অ্যাডভান্সড Snapdragon সিলিকন এবং Memory Tagging Extension (MTE) ব্যবহার করে বিশেষায়িত হার্ডওয়্যারের কো-ইঞ্জিনিয়ারিং করছি। GrapheneOS এখন পিক্সেলের মনোপলি থেকে বেরিয়ে এসে একটি বিশেষায়িত প্রাইভেসি এনক্লেভ থেকে রূপান্তরিত হয়ে এন্টারপ্রাইজ-গ্রেড অভয়ারণ্যে পরিণত হয়েছে।

এই অপারেটিং সিস্টেমটি একটি আপসহীন ক্রিপ্টোগ্রাফিক দুর্গ। এটি সম্পূর্ণভাবে এই নীতির ওপর কাজ করে যে, আপনার মোবাইল ডিভাইসটি শুধুমাত্র এর ব্যবহারকারীর জন্যই কাজ করবে এবং স্টেট-লেভেলের নজরদারি, কর্পোরেট ডেটা ব্রোকারেজ এবং অননুমোদিত ডেটা এক্সফিলট্রেশনের বিরুদ্ধে একটি অভেদ্য প্রাচীর হিসেবে দাঁড়িয়ে থাকবে।

দ্য ব্লুপ্রিন্ট অব রেসিস্ট্যান্স: এই বইটি কীভাবে ব্যবহার করবেন

GrapheneOS-এ পারদর্শী হওয়ার জন্য মোবাইল কম্পিউটিং নিয়ে আপনার মৌলিক ধারণাকে নতুনভাবে সাজানো প্রয়োজন। এটি সাধারণ ব্যবহারকারীদের কোনো কনজিউমার অ্যাপ্লায়েন্সের ইউজার ম্যানুয়াল নয়; এটি একটি হাই-সিকিউরিটি কমিউনিকেশন টার্মিনালের অপারেটর্স হ্যান্ডবুক। এই রূপান্তরটি সহজে বোঝার জন্য, বইটিকে প্রযুক্তিগত দক্ষতার ওপর ভিত্তি করে তিনটি ক্রমবর্ধমান স্তরে ভাগ করা হয়েছে। আপনি একজন প্রাইভেসি-সচেতন নাগরিক হোন, প্রতিকূল পরিবেশে কর্মরত একজন অনুসন্ধানী সাংবাদিক হোন, অথবা এন্টারপ্রাইজ ফ্লিট সুরক্ষায় নিয়োজিত কর্পোরেট সিকিউরিটি প্রফেশনাল হোন—এই বইটি আপনাকে ইন্সটলেশনের প্রাথমিক পর্যায় থেকে শুরু করে অপারেশনাল সিকিউরিটির (OpSec) সর্বোচ্চ চূড়া পর্যন্ত গাইড করবে।

পর্ব ১: ফাউন্ডেশন (The Basic Architecture)

এই বইটির প্রথম অংশটি স্ট্যান্ডার্ড মোবাইল প্যারাডাইমের ওপর আপনার নির্ভরশীলতা ভেঙে একটি সুরক্ষিত বেসলাইন তৈরি করার জন্য ডিজাইন করা হয়েছে। এখানে আমরা প্রাথমিক ও গুরুত্বপূর্ণ ধাপগুলো কভার করেছি: হার্ডওয়্যারের পূর্বশর্তগুলো বোঝা, নিরাপদে GrapheneOS ফার্মওয়্যার ফ্ল্যাশ করা এবং সিকিউর বুট প্রসেস নেভিগেট করা। আপনি শিখবেন কীভাবে হার্ডেনড অ্যাপ্লিকেশন স্যান্ডবক্স কনফিগার করতে হয়, গ্র্যানুলার সেন্সর পারমিশন পরিচালনা করতে হয় এবং ক্ষতিকারক

প্রোপাইটারি ফ্রেমওয়ার্কগুলোর ওপর নির্ভর না করেই একটি কার্যকরী ডেইলি-ড্রাইভার এনভায়রনমেন্ট স্থাপন করতে হয়। আমরা আপনাকে শেখাব কীভাবে Sandboxed Google Play নিরাপদে ইন্টিগ্রেট করতে হয়, যা আপনার প্রাইভেসির পরিধির সাথে আপস না করেই অ্যাপ কম্প্যাটিবিলিটি বজায় রাখতে সাহায্য করবে।

পর্ব ২: ফোর্ট্রেস (Advanced Hardening & Operational Security)

আপনার ভিত্তি সুরক্ষিত হওয়ার পর, আমরা অ্যাক্টিভ ডিফেন্সের স্তরে প্রবেশ করব। "ফোর্ট্রেস" পর্যায়টি আপনার ডিভাইসটিকে একটি সুরক্ষিত ফোন থেকে একটি অপারেশনাল দুর্গে উন্নীত করে। এই অংশটি নেটওয়ার্ক সার্ভিল্যান্স আর্কিটেকচারকে ভেঙে দেয় এবং রূপরেখা দেয় কীভাবে শক্তিশালী VPN টপোলজি বাস্তবায়ন করতে হয়, DNS লিক দূর করতে হয় এবং ক্ষতিকারক বেসব্যান্ড (Baseband) অ্যাটাক ও IMSI ক্যাচারের বিরুদ্ধে প্রতিরোধ গড়তে হয়। আপনি অ্যাডভান্সড প্রোফাইল আইসোলেশনে পারদর্শী হয়ে উঠবেন—যা মেটাডেটার ক্রস-কন্টামিনেশন রোধ করতে আপনার ডিজিটাল পরিচিতিকে আলাদা করে রাখবে। Physical Forensic Extraction-এর বিরুদ্ধে প্রতিরক্ষার জন্য প্রয়োজনীয় OpSec নীতিগুলো আপনি শিখবেন, যেখানে ডিভাইস অফলাইন থাকা অবস্থায় ডেটা সুরক্ষিত রাখতে অটো-রিবুট প্রোটোকল এবং Weaver API থ্রটলিং ব্যবহার করা হবে।

পর্ব ৩: ব্যাটলফিল্ড (Professional Defense & Zero-Day Mitigation)

সর্বশেষ স্তরটি তাদের জন্য সংরক্ষিত যারা সর্বোচ্চ থ্রেট মডেলের অধীনে কাজ করেন। "ব্যাটলফিল্ড" পর্বটি GrapheneOS-এর টেকনিক্যাল আর্কিটেকচারের সর্বাধুনিক স্তর নিয়ে আলোচনা করে। এখানে আমরা `hardened_malloc`, হার্ডওয়্যার মেমরি ট্যাগিং (MTE), এবং কার্নেল পেজ-টেবিল আইসোলেশনের কার্যপ্রণালী বিশ্লেষণ করব। আপনি বুঝতে পারবেন কীভাবে GrapheneOS মেমরি করাপশন বাগ প্রশমিত করে এবং অত্যাধুনিক, রাষ্ট্র-পরিচালিত Zero-day exploit চেইনগুলোকে নিষ্ক্রিয় করে। আমরা এন্টারপ্রাইজ ফ্লিট ডিপ্লয়মেন্ট নিয়েও আলোচনা করব, যেখানে বিশদভাবে দেখানো হবে কীভাবে ২০২৬ সালের Motorola অংশীদারত্ব প্রাতিষ্ঠানিক ডিভাইসগুলোর জিরো-ট্রাস্ট ম্যানেজমেন্ট সক্ষম করে। এটি হলো প্রফেশনাল-গ্রেড সাইবার ওয়ারফেয়ার, যা এমন অপারেটরদের জন্য ডিজাইন করা হয়েছে যারা নিরাপত্তায় বিন্দুমাত্র ছাড় দিতে পারেন না।

আপনার সিদ্ধান্ত

নজরদারির শিকার হওয়াটা এখন আপনার নিজের ইচ্ছার ওপর নির্ভর করে। গত এক দশক ধরে, এই সিদ্ধান্তটি আপনার হয়ে নিয়েছিল বিশাল টেকনোলজিকাল কর্পোরেশনগুলো, যারা আপনার জীবনকে একটি কাঁচামাল হিসেবে বিবেচনা করেছে যা মাইন, প্রসেস এবং বিক্রি করা যায়। কিন্তু GrapheneOS-এর ব্যবহার সেই নিয়ন্ত্রণ পুনরায় আপনার হাতেই ফিরিয়ে দেয়।

আপনি যে ডিভাইসটি তৈরি করতে যাচ্ছেন, তা আপনাকে ট্র্যাক করবে না। এটি আপনার দৈনন্দিন অভ্যাসগুলোকে অর্থে রূপান্তর করবে না। ফরেনসিক জব্দের সময় এটি তার ক্রিপ্টোগ্রাফিক গোপনীয়তাগুলো প্রকাশ করবে না। আপনি একে ঠিক যা নির্দেশ দেবেন, এটি কেবল তা-ই করবে, এবং এর বাইরে বিন্দুমাত্র কিছুই করবে না।

ডিজিটাল প্যানোপটিকন হলো বিশাল, ব্যাপকভাবে অর্থায়িত এবং দৃশ্যত এড়িয়ে যাওয়ার অযোগ্য। তবে এর দেয়ালগুলো সফটওয়্যার দিয়ে তৈরি, আর সফটওয়্যার নতুন করে লেখা যায়। পৃষ্ঠা উল্টান, এবং চলুন আপনার দুর্গ নির্মাণের কাজ শুরু করা যাক।

পর্ব ১: ফাউন্ডেশন

অধ্যায় ১: ভূমিকা

ফাউন্ডেশনে আপনাকে স্বাগতম। অ্যাডভান্সড নেটওয়ার্ক ডিফেন্স ডিপ্লয় করা কিংবা Physical Forensic Extraction-এর বিরুদ্ধে প্রতিরোধ গড়ার আগে, আমাদেরকে অবশ্যই একটি শক্ত ভিত তৈরি করতে হবে। এই প্রথম পর্বের মূল উদ্দেশ্য হলো, ডিফল্টভাবে আপস করা স্ট্যান্ডার্ড মোবাইল প্যারাডাইমের ওপর থেকে আপনার নির্ভরতা সম্পূর্ণ ভেঙে ফেলা এবং একটি অত্যন্ত সুরক্ষিত ও কার্যকরী বেসলাইন তৈরি করা।

অধিকাংশ ব্যবহারকারীর জন্যই একটি হাই-সিকিউরিটি অপারেটিং সিস্টেমে ট্রানজিশন করাটা প্রযুক্তির সাথে তাদের দৈনন্দিন ইন্টারঅ্যাকশনের ক্ষেত্রে একটি বড় ধরনের পরিবর্তন। এর জন্য সার্ভিল্যান্স ক্যাপিটালিজমের (নজরদারি পুঁজিবাদ) দেয়া তথাকথিত সুবিধাগুলোকে ভুলে যেতে হয়—যেখানে আপনার সম্পূর্ণ ব্যক্তিগত তথ্যের বিনিময়ে আপনাকে কিছু ফ্রি অ্যাপ্লিকেশন ব্যবহারের সুযোগ দেওয়া হয়। এই প্রথম পর্বে, আমরা আপনাকে ধাপে ধাপে এই ট্রানজিশন প্রক্রিয়ার মধ্য দিয়ে নিয়ে যাব। আমরা এই অপারেটিং সিস্টেমের তাত্ত্বিক আর্কিটেকচার সম্পর্কে জানব, ঠিক কোন ধরনের হুমকির বিরুদ্ধে এটি আপনাকে সুরক্ষা দেয় তা বুঝব, এবং আপনার ডিজিটাল জীবনকে একটি সুরক্ষিত এনক্লেভে স্থানান্তরিত করার জন্য অতি প্রয়োজনীয় প্রথম পদক্ষেপগুলোর রূপরেখা তৈরি করব।

এখান থেকেই আপনি আপনার ডিজিটাল সার্বভৌমত্ব (Digital Sovereignty) পুনরুদ্ধার করতে শুরু করবেন। এবং এই প্রক্রিয়াটি শুরু হবে আমাদের হাতে থাকা এই হাতিয়ারের প্রকৃত রূপটি সঠিকভাবে অনুধাবন করার মাধ্যমে।

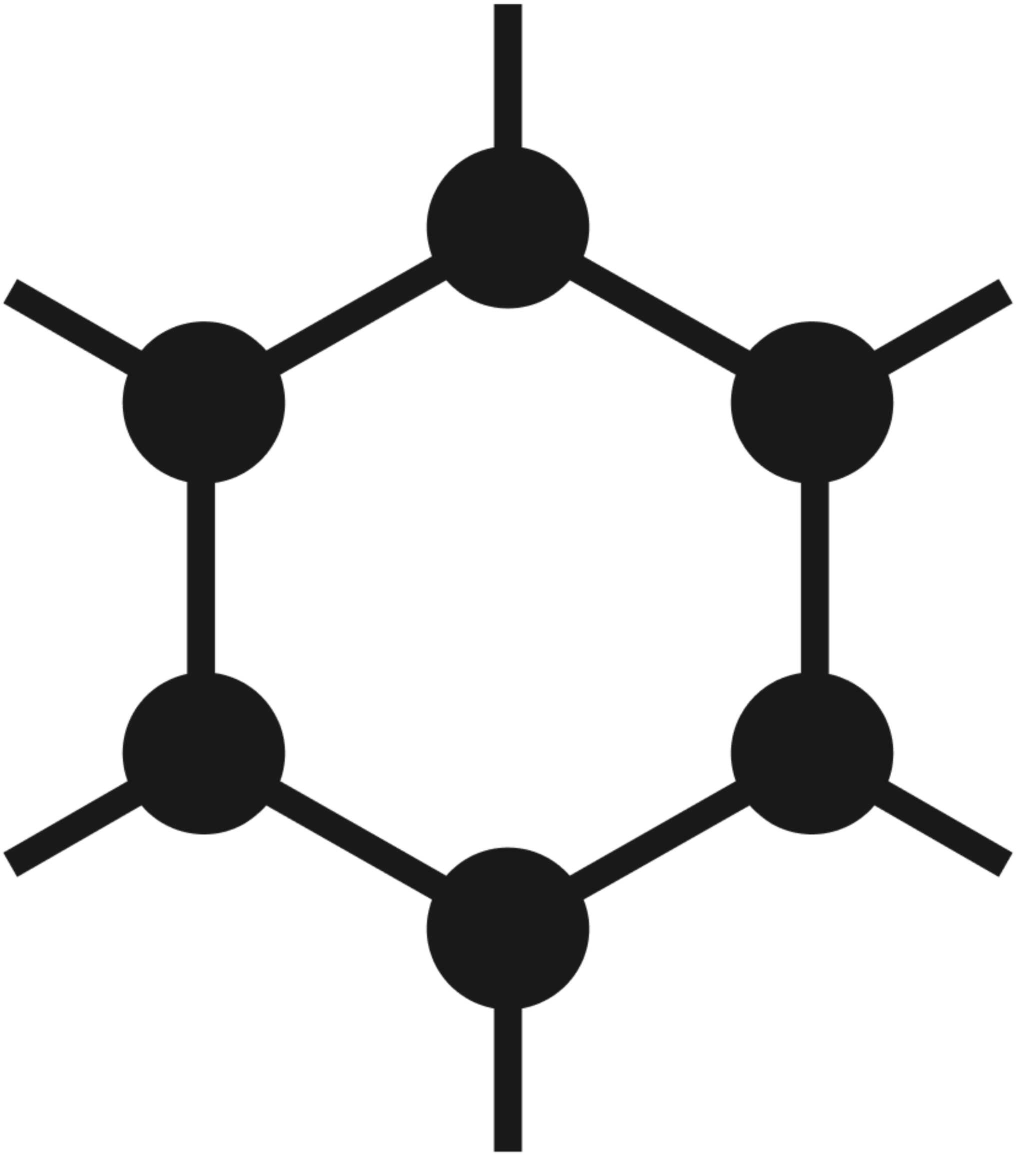
১.১ GrapheneOS কী?

GrapheneOS-কে বুঝতে হলে, স্ট্যান্ডার্ড কমার্শিয়াল স্মার্টফোনগুলোর আর্কিটেকচারটি কল্পনা করা বেশ সহায়ক হতে পারে।

ধরে নিন, স্ট্যান্ডার্ড Android বা Apple-এর iOS হলো সম্পূর্ণ কাঁচের দেয়াল দিয়ে ঘেরা একটি বিশাল ও সুদৃশ্য বাড়ি। এটি দেখতে চমৎকার, বাইরের জগতের সাথে এর চমৎকার সংযোগ রয়েছে এবং এটি অত্যন্ত সুবিধাজনক। কিন্তু সমস্যা হলো, এর দেয়ালগুলো স্বচ্ছ হওয়ায় যে কেউ বাইরে থেকে উঁকি দিতে পারে—আপনার বাড়িওয়ালা (Google বা Apple), ইউটিলিটি

কোম্পানি (আপনার টেলিকম ক্যারিয়ার), এবং অসংখ্য সেলসম্যান (থার্ড-পার্টি অ্যাপ্লিকেশন)—এরা সবাই অনায়াসেই দেখতে পায় আপনি ভেতরে কী করছেন। আপনি কখন ঘুম থেকে ওঠেন, কার সাথে কথা বলেন, এবং আপনি কী কী কনজিউম করেন, তার সবকিছুই তারা দেখতে পায়।

যখনই প্রাইভেসি নিয়ে কোনো বিতর্কের সৃষ্টি হয়, তখন এই কোম্পানিগুলো পাতলা পর্দার মতো কিছু "সমাধান" নিয়ে হাজির হয়। তারা আপনাকে সেটিংস মেনুতে এমন কিছু অপশন দেয়, যার মাধ্যমে আপনি অ্যাপ্লিকেশনগুলোকে আপনাকে ট্র্যাক না করার জন্য বিনীত অনুরোধ করতে পারেন। কিন্তু একটি পর্দা কখনোই পুরো বাড়ির আর্কিটেকচার বদলে দিতে পারে না। এর অন্তর্নিহিত কাঠামোটি তখনও মৌলিকভাবেই স্বচ্ছ থেকে যায়। Telemetry ইঞ্জিনগুলো তখনও ব্যাকগ্রাউন্ডে চলতে থাকে, ব্যাকগ্রাউন্ড অ্যানালিটিক্স তখনও কোম্পানির সার্ভারে রিপোর্ট পাঠায়, এবং অত্যাধুনিক অনুপ্রবেশকারীরা (Sophisticated intruders) তখনও খুব সহজেই সেই কাঁচ ভেঙে ভেতরে প্রবেশ করতে পারে।



GrapheneOS কোনো পাতলা পর্দা নয়। GrapheneOS হলো একটি রিইনফোর্সড কংক্রিট বাঙ্কার।

এটি একদম বেয়ার মেটাল (bedrock) স্তর থেকে সম্পূর্ণ স্ট্রাকচারাল ইন্টিগ্রিটির সাথে ডিজাইন করা হয়েছে। এখানে কোনো কাঁচের দেয়াল নেই। এই কাঠামোর ডিফল্ট স্টেটই হলো সম্পূর্ণ অস্বচ্ছতা (opacity)। ব্যবহারকারী হিসেবে এই বাঙ্কারের সম্পূর্ণ নিয়ন্ত্রণ ও সার্বভৌমত্ব কেবল আপনার হাতেই। যদি কোনো অ্যাপ্লিকেশন আপনার ক্যামেরা, মাইক্রোফোন বা বাইরের

নেটওয়ার্কে অ্যাক্সেস চায়, তবে সেটি কেবল কাঁচের ভেতর দিয়ে উঁকি দিতে পারবে না। তাকে স্পষ্টভাবে আপনার কাছে চাবি চাইতে হবে, এবং সেই চাবি দেওয়ার বা যেকোনো মুহূর্তে তা বাতিল করার চূড়ান্ত ক্ষমতা কেবল আপনারই থাকবে, একদম গ্র্যানুলার লেভেল পর্যন্ত।

আর্কিটেকচারের উৎপত্তি

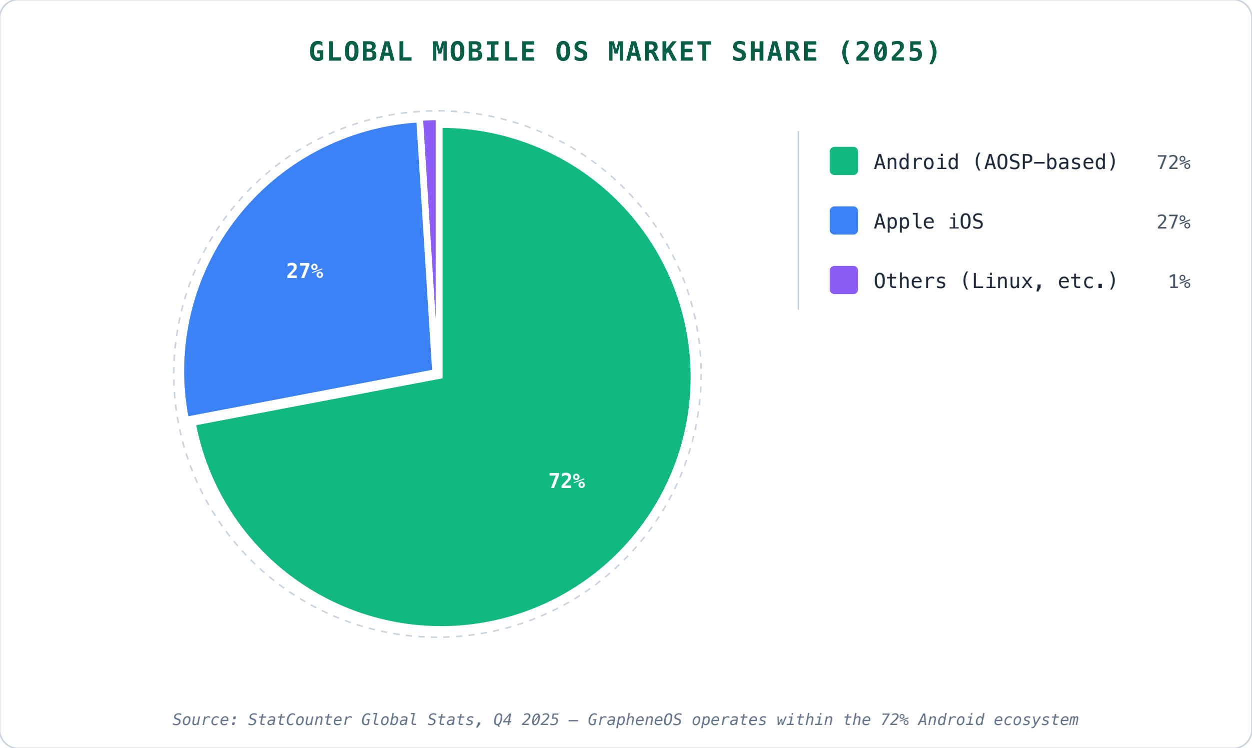
এই আপসহীন আর্কিটেকচারটি রাতারাতি তৈরি হয়নি। ২০১৪ সালের শেষের দিকে সিকিউরিটি রিসার্চার ড্যানিয়েল মিকের (Daniel Micay) হাত ধরে GrapheneOS-এর যাত্রা শুরু হয়। প্রাথমিকভাবে এটি একটি এক্সপেরিমেন্টাল ওপেন-সোর্স প্রজেক্ট হিসেবে শুরু হলেও, মিকের মূল লক্ষ্য ছিল Android অপারেটিং সিস্টেমকে একদম ভিত্তিমূল থেকে সুপরিকল্পিতভাবে শক্তিশালী (fortify) করা।

সেই সময়ে মোবাইল সাইবার সিকিউরিটির ল্যান্ডস্কেপ মূলত অ্যাপ্লিকেশন লেয়ারের ওপর নির্ভরশীল ছিল—যেমন স্ট্যান্ডার্ড Android-এর ওপর ভিত্তি করে এনক্রিপ্টেড মেসেজিং অ্যাপ বা VPN তৈরি করা। মিকাই বুঝতে পেরেছিলেন যে এই পদ্ধতিটি একটি ভুল অ্যাপ্রোচ। তার দর্শন ছিল, প্রকৃত নিরাপত্তা নিশ্চিত করতে হলে সেটিকে অবশ্যই অপারেটিং সিস্টেমের একেবারে সর্বনিম্ন স্তরে সরাসরি ইঞ্জিনিয়ার করতে হবে। পরবর্তী এক দশকে, বিভিন্ন ধাপে এবং কঠোর টেকনিক্যাল বিবর্তনের মাধ্যমে, এই প্রজেক্টটিকে বর্তমান রূপ দেওয়া হয়েছে: একটি স্বাধীন অপারেটিং সিস্টেম, যা বাজারজাতকরণের (marketability) খাতিরে কখনোই নিরাপত্তার সাথে আপস করে না।

এই অপারেটিং সিস্টেমটি কঠোরভাবে একটি জিরো-টেলিমেট্রি (zero-telemetry) এনভায়রনমেন্ট নিশ্চিত করে। ইন্সটল করার পরপরই (out of the box), এটি বাইরের কোনো কর্পোরেট সার্ভারের সাথে বিন্দুমাত্র ব্যাকগ্রাউন্ড কানেকশন তৈরি করে না। এটি ডায়াগনস্টিক ডেটা সংগ্রহ করে না, ব্যবহারকারীর কোনো ডেমোগ্রাফিক প্রোফাইল তৈরি করে না, এবং আধুনিক মোবাইল ইন্ডাস্ট্রির সংজ্ঞায়িত সেই সর্বব্যাপী ট্র্যাকিং পরিকাঠামোকে এটি সম্পূর্ণভাবে প্রত্যাখ্যান করে।

> 💡 ***Pro Tip: প্রাতিষ্ঠানিক অখণ্ডতা (Institutional Integrity)*** *GrapheneOS-এর ডেভেলপমেন্ট প্রক্রিয়া সম্পূর্ণভাবে কানাডায় নিবন্ধিত একটি অলাভজনক সংস্থা, GrapheneOS Foundation দ্বারা পরিচালিত হয়। এই কাঠামোগত স্বাধীনতা নিশ্চিত করে যে, প্রজেক্টটি কর্পোরেট প্রভাব, ডেটা মনিটাইজেশনের লোভ বা প্রোপাইটারি লক-ইনের মতো বিষয়গুলো থেকে সম্পূর্ণ মুক্ত। তাদের একমাত্র ম্যান্ডেট হলো এন্ড-ইউজারের (end-user) নিরঙ্কুশ নিরাপত্তা।*

De-Googling বনাম Hardening: গুরুত্বপূর্ণ পার্থক্য



প্রাইভেসি-ফোকাসড অপারেটিং সিস্টেম নিয়ে রিসার্চ করার সময় নতুন ব্যবহারকারীরা প্রায়শই দুটি শব্দ দেখতে পান যা একে অপরের পরিপূরক হিসেবে ব্যবহৃত হয়: "De-Googling" এবং "Hardening"। এই দুটি ধারণার মধ্যকার মূল পার্থক্য বুঝতে পারাটা অত্যন্ত গুরুত্বপূর্ণ, কারণ এটি আপনাকে বোঝাবে কেন GrapheneOS এই ইন্ডাস্ট্রিতে অনন্য।

De-Googling বলতে বোঝায় Android অপারেটিং সিস্টেম থেকে Google-এর প্রোপাইটারি অ্যাপ্লিকেশন এবং ব্যাকএন্ড সার্ভিসগুলো (যেমন Google Play Services) সম্পূর্ণভাবে মুছে ফেলার প্রক্রিয়া। De-Googling-এর প্রধান লক্ষ্য হলো প্রাইভেসি নিশ্চিত করা। এই সার্ভিসগুলো সরিয়ে ফেলার মাধ্যমে আপনি আপনার ডিভাইস থেকে Google-এর সার্ভারে প্রতিনিয়ত যে টেলিমেট্রি ডেটা প্রবাহিত হয়, তা স্থায়ীভাবে বন্ধ করতে পারেন। অনেক জনপ্রিয় কাস্টম রমই (Custom ROMs) মূলত De-Googling-এর ওপর ফোকাস করে তৈরি। এগুলো কর্পোরেট সার্ভিল্যান্সমুক্ত একটি চমৎকার অভিজ্ঞতা দেয়, যা প্রাইভেসির ক্ষেত্রে একটি বিশাল অর্জন।

তবে, শুধুমাত্র De-Googling একটি ফোনকে কখনোই *সুরক্ষিত (secure)* করে তোলে না। আপনি যদি একটি ফোন থেকে Google-কে সরিয়ে দেন, তবে এটি আর আপনাকে ট্র্যাক করবে না ঠিকই, কিন্তু বাহ্যিক আক্রমণকারীদের বিরুদ্ধে প্রতিরোধ গড়ার ক্ষেত্রে ডিভাইসটি গঠনগত দিক দিয়ে তখনো একটি স্ট্যান্ডার্ড Android ফোনের মতোই দুর্বল থেকে যায়। এটি তখনো

মেমরি করাপশন বাগ, Baseband এক্সপ্লয়েট এবং আইনপ্রয়োগকারী সংস্থা বা থ্রেট অ্যাক্টরদের দ্বারা পরিচালিত Physical Forensic Extraction-এর সামনে অরক্ষিত থাকে।

অন্যদিকে, **Hardening** হলো অপারেটিং সিস্টেমটিকে শারীরিকভাবে এবং গাণিতিকভাবে এমনভাবে শক্তিশালী (reinforce) করার প্রক্রিয়া, যা যেকোনো অ্যাক্টিভ ও টার্গেটেড অ্যাটাক প্রতিহত করতে সক্ষম। Hardening-এর মূল লক্ষ্য হলো অ্যাটাক সারফেস (attack surface) উল্লেখযোগ্যভাবে হ্রাস করা। এর মধ্যে রয়েছে বাফার ওভারফ্লো (buffer overflow) অ্যাটাক প্রতিরোধ করার জন্য মেমরি অ্যালোকেটরকে নতুন করে লেখা, কার্নেলকে (kernel) গভীরভাবে আইসোলেট করা, অ্যাপ্লিকেশনগুলোর জন্য অ্যাডভান্সড Sandboxing প্রোটোকল বাস্তবায়ন করা, এবং ডেটা অ্যাট রেস্ট (data at rest) সুরক্ষিত করার জন্য হার্ডওয়্যার-লেভেলের এনক্রিপশন ব্যবহার করা।

GrapheneOS-এর মূল শক্তি হলো এটি এই **দুটি কাজই একসাথে** করে।

GrapheneOS সম্পূর্ণ De-Googled এনভায়রনমেন্টের মাধ্যমে আপনার নিরঙ্কুশ প্রাইভেসি নিশ্চিত করে, আর ঠিক একইসাথে এটি কমার্শিয়াল সেক্টরে উপলব্ধ সবচেয়ে অ্যাগ্রেসিভ Hardening টেকনিকগুলোও প্রয়োগ করে। এটি শুধু আপনাকে বড় বড় টেক কোম্পানিগুলোর নজরদারি থেকেই বাঁচায় না; বরং এটি রাষ্ট্র-পরিচালিত Zero-day exploit চেইন, Sandboxing ভেঙে বেরিয়ে আসার চেষ্টায় থাকা ম্যালিশিয়াস অ্যাপ্লিকেশন, এবং অত্যাধুনিক ফরেনসিক ডেটা এক্সট্রাকশন টুলগুলোর বিরুদ্ধেও আপনার ডিভাইসকে সক্রিয়ভাবে রক্ষা করে।

যেখানে অন্যান্য কাস্টম রমগুলো শুধুমাত্র ট্র্যাকিং সফটওয়্যার সরিয়ে ফেলার দিকে মনোযোগ দেয়, সেখানে GrapheneOS ডিভাইসের সম্পূর্ণ ভিত্তিটাকেই নতুন করে তৈরি করে। এটি মূলত এই নীতির ওপর ভিত্তি করে কাজ করে যে প্রাইভেসি এবং সিকিউরিটি অঙ্গাঙ্গিভাবে জড়িত: দুর্ভেদ্য সিকিউরিটি ছাড়া আপনি কখনোই সত্যিকারের প্রাইভেসি পেতে পারেন না, আবার অপারেটিং সিস্টেমটি যদি আপনার ওপর নজরদারি করার জন্যই ডিজাইন করা হয়, তবে আপনি কখনোই নিশ্ছিদ্র সিকিউরিটি পাবেন না।

GrapheneOS ইন্সটল করার মাধ্যমে আপনি শুধু সার্ভিল্যান্স ইকোনমি থেকেই বের হয়ে আসছেন না; আপনি এর বিরুদ্ধে এন্টারপ্রাইজ-গ্রেড ডিফেন্সিভ কাউন্টারমেজার্স (defensive countermeasures) ডিপ্লয় করছেন। পরবর্তী সেকশনগুলোতে, আমরা এই সিস্টেমের মেকানিজমগুলো ধাপে ধাপে আলোচনা করব, যাতে আপনার এই নিরাপত্তার ভিত্তিটি সম্পূর্ণ ত্রুটিমুক্তভাবে তৈরি হয়।

1.2 Hardware Requirements এবং Pixel Exclusivity

যখন privacy কমিউনিটির নতুন ব্যবহারকারীরা প্রথমবারের মতো GrapheneOS নিয়ে রিসার্চ শুরু করেন, তখন তারা প্রায় সাথে সাথেই একটি বিশাল এবং দৃশ্যত সাংঘর্ষিক প্রতিবন্ধকতার সম্মুখীন হন। তাদেরকে বলা হয় যে, চূড়ান্ত anti-surveillance এবং de-Googled স্মার্টফোন তৈরি করতে হলে, প্রথমেই তাদেরকে এমন একটি স্মার্টফোন কিনতে হবে যা স্বয়ং Google-এর তৈরি।

অনেকের কাছেই এটিকে একটি ফাঁদ বলে মনে হয়। এই cognitive dissonance বা মানসিক দ্বন্দ্বটি পুরোপুরি বোধগম্য। বিশ্বের সবচেয়ে উন্নত privacy অপারেটিং সিস্টেম কীভাবে এমন একটি hardware-এর ওপর এককভাবে নির্ভর করতে পারে, যা বিশ্বের সবচেয়ে বিস্তৃত ডেটা ব্রোকারেজ কর্পোরেশন দ্বারা উৎপাদিত? একজন সাধারণ পর্যবেক্ষকের চোখে, Google-কে এড়ানোর জন্য একটি Google Pixel কেনাটা অনেকটা পেট্রোল দিয়ে আগুন নেভানোর চেষ্টার মতোই শোনায়।

তবে, deep systems engineering-এর জগতে এটি কোনো বৈপরীত্য বা contradiction নয়; এটি একটি সুপরিকল্পিত এবং অত্যন্ত টেকনিক্যাল প্রয়োজনীয়তা। GrapheneOS-কে মাস্টার করতে হলে, আপনাকে প্রথমেই hardware এবং software-এর মধ্যকার কঠোর বিভাজনটি বুঝতে হবে এবং জানতে হবে কেন প্রায় এক দশক ধরে Pixel-ই ছিল একমাত্র নিরাপদ আশ্রয়স্থল (sanctuary), যা এই বাঙ্কারকে সুরক্ষিত রাখতে সক্ষম ছিল।

The Great Irony: Silicon থেকে Software-কে আলাদা করা

এই বিভ্রান্তির উৎপত্তি হয় স্মার্টফোনকে একটি একক ও অবিভাজ্য সত্তা হিসেবে বিবেচনা করা থেকে। বাস্তবে এটি তা নয়। একটি আধুনিক স্মার্টফোন হলো অত্যন্ত উন্নত ফিজিক্যাল hardware (silicon, মেমরি, secure elements) এবং অপারেটিং সিস্টেমের (software) একটি যৌগিক কাঠামো, যা এই hardware-কে নিয়ন্ত্রণ করে।

Google-এর কর্পোরেট স্ট্রাকচারও এই বিভাজনকেই প্রতিফলিত করে। Google-এর hardware ডিভিশন এই গ্রহের সবচেয়ে শক্তিশালী, physically secure এবং cryptographically উন্নত consumer silicon ডিজাইন করে। তারা অবিশ্বাস্য রকমের সামরিক-গ্রেডের ভল্ট তৈরি করে। বিপরীতদিকে, Google-এর software ডিভিশন স্ট্যান্ডার্ড Android অপারেটিং সিস্টেম লেখে—যা মূলত একটি ব্যাপক telemetry ইঞ্জিন। এটি সেই ভল্টের ভেতরে কাজ করার জন্য ডিজাইন করা হয়েছে, যা আপনার প্রতিটি কাজকে ক্যাটালগ করে এবং তাদের নিজস্ব সার্ভারগুলোতে পাঠিয়ে দেয়।

আপনি যখন একটি স্ট্যান্ডার্ড Google Pixel কেনেন, তখন আপনি মূলত একটি চমৎকার ফিজিক্যাল ভল্ট কিনছেন, যার মধ্যে আগে থেকেই একটি আক্রমণাত্মক surveillance ইঞ্জিন লোড করা আছে। GrapheneOS-এর ম্যাজিকটি হলো এই সম্পর্ককে সম্পূর্ণভাবে ছিন্ন করার সক্ষমতার মধ্যে।

Installation প্রসেসের সময়, আপনি গাণিতিকভাবে ডিভাইস থেকে Google-এর software-কে ধ্বংস করে দেন। আপনি telemetry, ট্র্যাকিং এবং proprietary framework-গুলোকে পুরোপুরি মুছে ফেলেন। এরপর যা অবশিষ্ট থাকে, তা হলো একটি অবিশ্বাস্য রকমের সুরক্ষিত, খালি ফিজিক্যাল ভল্ট। GrapheneOS এরপর সেই ভল্টের দখল নেয় এবং secure elements, cryptographic keys এবং bootloader-এর ওপর সম্পূর্ণ নিয়ন্ত্রণ প্রতিষ্ঠা করে। আপনি মূলত Google-এর software-এর বিরুদ্ধেই Google-এর hardware-কে অস্ত্র হিসেবে ব্যবহার করছেন।

ঐতিহাসিকভাবে শুধুমাত্র Pixel কেন?

প্রজেক্টটির অস্তিত্বের প্রথম দশ বছরে, GrapheneOS আনুষ্ঠানিকভাবে শুধুমাত্র Google Pixel ডিভাইসেই উপলব্ধ ছিল। এই exclusivity বা একচেটিয়া অধিকার কোনো ডেভেলপারের পছন্দ বা ব্র্যান্ড আনুগত্যের কারণে তৈরি হয়নি; বরং এটি ছিল আপোসহীন cryptographic সিকিউরিটির একটি ম্যান্ডেট। GrapheneOS রান করার জন্য, একটি ডিভাইসকে অত্যন্ত কঠোর hardware requirement-এর চেকলিস্ট পূরণ করতে হয়, যা ঐতিহাসিকভাবে অন্যান্য Original Equipment Manufacturers (OEMs) প্রদান করতে চরমভাবে ব্যর্থ হয়েছে।

ঠিক এই কারণেই Pixel এত দীর্ঘ সময় ধরে একা দাঁড়িয়ে ছিল:

1. Proper OEM Unlocking এবং Relocking বেশিরভাগ ম্যানুফ্যাকচারার—যেমন Samsung, Huawei, বা টেলিকম ক্যারিয়ারের মাধ্যমে বিক্রি হওয়া ডিভাইসগুলো—তাদের ব্যবহারকারীদের চরম সন্দেহের চোখে দেখে। তারা bootloader-কে শারীরিকভাবে লক করে রাখে, যাতে ব্যবহারকারীরা ভিন্ন কোনো অপারেটিং সিস্টেম ইনস্টল করতে না পারে। তারা proprietary ইকোসিস্টেমের লক-ইন বলবৎ করে এবং custom firmware ফ্ল্যাশ করার যেকোনো চেষ্টাকে একটি ভয়ংকর সিকিউরিটি ব্রিচ হিসেবে বিবেচনা করে। সরাসরি কেনা এবং আনলক করা Google Pixel-গুলো ব্যবহারকারীকে আনুষ্ঠানিকভাবে bootloader আনলক করার অনুমতি দেয়। সবচেয়ে গুরুত্বপূর্ণ বিষয়টি হলো, Pixel ব্যবহারকারীদের একটি custom OS ফ্ল্যাশ করার সুযোগ দেয় এবং এরপর custom cryptographic signing keys ব্যবহার করে bootloader-কে পুনরায় *relock* করার সুবিধাও প্রদান করে।

2. Verified Boot with User Keys আপনি যখন একটি স্মার্টফোন চালু করেন, তখন আপনার এই চূড়ান্ত নিশ্চয়তার প্রয়োজন হয় যে, আপনি যখন ঘুমাচ্ছিলেন তখন কোনো ম্যালওয়্যার বা একটি "evil maid" অ্যাটাকের মাধ্যমে গোপনে আপনার অপারেটিং সিস্টেমের সাথে টেম্পারিং করা হয়নি। Android Verified Boot (AVB) এই গাণিতিক গ্যারান্টি প্রদান করে।

যদিও অনেক ফোন AVB ব্যবহার করে, তারা শুধুমাত্র ম্যানুফ্যাকচারারের key-গুলোই গ্রহণ করে। কিন্তু Pixel-গুলোকে এমনভাবে ইঞ্জিনিয়ারিং করা হয়েছে যেন তারা একটি custom অপারেটিং সিস্টেমের key ব্যবহার করে AVB সাপোর্ট করতে পারে। যখন একটি GrapheneOS ডিভাইস বুট হয়, তখন hardware শারীরিকভাবে OS-এর cryptographic signature ভেরিফাই করে, যা নিশ্চিত করে যে এটি GrapheneOS-এর একটি প্রিস্টিন এবং আনটেম্পারড (untampered) বিল্ড। যদি কোডের একটি লাইনও পরিবর্তন করা হয়, তবে ডিভাইসটি বুট হতে অস্বীকার করবে।

3. The Titan M2 Secure Enclave একটি Pixel-এর মাদারবোর্ডে Titan M2 (বা পুরনো মডেলগুলোতে মূল Titan M) নামক একটি শারীরিকভাবে বিচ্ছিন্ন, ডেডিকেটেড সিকিউরিটি চিপ থাকে। এই secure element-টি চূড়ান্ত root of trust হিসেবে কাজ করে। এটি মূল প্রসেসর থেকে সম্পূর্ণ বিচ্ছিন্ন অবস্থায় আপনার এনক্রিপশন key-গুলোকে নিরাপদে সংরক্ষণ করে। সবচেয়ে সমালোচনামূলক বিষয়টি হলো, এটি Weaver API সাপোর্ট করে, যা পাসকোড দেওয়ার প্রচেষ্টায় hardware-backed থ্রটলিং (throttling) প্রদান করে। যদি কোনো ক্ষতিকর হ্যাকার বা ফরেনসিক এক্সট্রাকশন টুল আপনার পাসওয়ার্ড brute-force করার চেষ্টা করে, তবে Titan M2 চিপ শারীরিকভাবে তাদের গেস বা অনুমানের গতিকে সীমিত করে দেয়, যার ফলে যে প্রক্রিয়াটিতে কয়েক ঘণ্টা লাগার কথা, তা শতাব্দীকালের একটি প্রচেষ্টায় পরিণত হয়।

Supported Pixel Device Tiers

যেহেতু GrapheneOS উন্নত hardware ফিচারের ওপর ব্যাপকভাবে নির্ভর করে, তাই এটি কঠোরভাবে hardware lifecycle মেনে চলে। যখন একটি ডিভাইস ম্যানুফ্যাকচারারের কাছ থেকে আর কোনো underlying ফার্মওয়্যার আপডেট পায় না, তখন সেটি baseband বা hardware-level এক্সপ্লয়েটের বিরুদ্ধে আর সত্যিকার অর্থে সুরক্ষিত থাকতে পারে না, এবং GrapheneOS তখন সেটির সাপোর্ট বন্ধ করে দেয়।

বর্তমানে, ইকোসিস্টেমটি Pixel আর্কিটেকচারের নিম্নলিখিত টিয়ারগুলোকে সাপোর্ট করে:

DEVICE SERIES	SOC	SECURITY TIER	KEY CAPABILITY	MTE SUPPORT
Pixel 6	Tensor G1	Baseline	First custom Google silicon; Titan M2	No
Pixel 7	Tensor G2	Mid-Tier	Refined Tensor arch; stable Titan M2	No
Pixel 8	Tensor G3	Advanced	Hardware Memory Tagging Extensions	**Yes**
Pixel 9	Tensor G4	Advanced+	Isolated on-device AI; upgraded modem	**Yes**
Pixel 10	Tensor G5	Pinnacle	TSMC custom arch; fastest MTE execution	**Yes**

Pixel 6 সিরিজটি চূড়ান্ত বেসলাইনের প্রতিনিধিত্ব করে—যা Google-এর কাস্টম Tensor silicon প্রবর্তনের জন্য যুগান্তকারী ছিল, কিন্তু এক্সটেন্ডেড সিকিউরিটি আপডেটের জন্য এখন তা এন্ড-অফ-লাইফ (end-of-life)-এর কাছাকাছি চলে এসেছে। Pixel 7 সিরিজ Titan M2 secure enclave-এর একটি অত্যন্ত নির্ভরযোগ্য এবং পরিমার্জিত ইমপ্লিমেন্টেশন অফার করে। Pixel 8 ছিল একটি যুগান্তকারী লাফ বা monumental leap, যা Hardware MTE প্রবর্তন করে—এটি এমন একটি ফিজিক্যাল silicon ফিচার যা hardware লেভেলেই বিশাল শ্রেণির মেমরি করাপশন ভালনারেবিলিটিকে নিষ্ক্রিয় করে দেয়। Pixel 9 নিয়ে আসে উন্নত on-device AI silicon, যাকে GrapheneOS ক্লাউড telemetry থেকে সম্পূর্ণ আলাদা করে রাখে, সেইসাথে উল্লেখযোগ্যভাবে আপগ্রেড করা মডেম আর্কিটেকচার প্রদান করে। Pixel 10, যা সম্পূর্ণরূপে কাস্টমাইজড TSMC-ম্যানুফ্যাকচারড আর্কিটেকচারে স্থানান্তরিত হয়েছে, এটি হাই-রিস্ক অপারেটরদের জন্য চূড়ান্ত দুর্গের (ultimate fortress) প্রতিনিধিত্ব করে।

The 2026 Milestone: একচেটিয়া আধিপত্য ভেঙে দেওয়া

যদিও Google-এর hardware-কে কাজে লাগানো একটি ইঞ্জিনিয়ারিং মাস্টারস্ট্রোক ছিল, কিন্তু Pixel-এর একচেটিয়া আধিপত্য একটি বিশাল ও দীর্ঘমেয়াদী কৌশলগত ভালনারেবিলিটি তৈরি করেছিল। সাইবার সিকিউরিটির প্রবাদবাক্য "One is none, two is one" এই পজিশনের ভঙ্গুরতাকে ব্যাপকভাবে আন্ডারস্কোর করেছে। যদি Google হঠাৎ করে তাদের ভবিষ্যতের কোনো Pixel জেনারেশনে bootloader পলিসি পরিবর্তন করে বসে—হয়তো কোনো সরকার বা টেলিকম ক্যারিয়ারের চাপে পড়ে—তবে GrapheneOS প্রজেক্টটি একটি কার্যকর hardware প্ল্যাটফর্ম ছাড়াই মুখ থুবড়ে পড়বে, যা বিশ্বব্যাপী লাখো ব্যবহারকারীর সুরক্ষিত যোগাযোগ ব্যবস্থাকে হুমকির মুখে ফেলবে।

Pixel-এর এই মনোপলি বা একচেটিয়া আধিপত্য ভাঙাটি প্রজেক্টের সবচেয়ে জরুরি উদ্দেশ্য হয়ে দাঁড়ায়। অবশেষে ২০২৫ সালের শেষ দিকে এবং ২০২৬ সালের শুরুর দিকে এই প্যারাডাইমটি সম্পূর্ণভাবে পরিবর্তিত হয়।

২০২৬ সালে Mobile World Congress (MWC)-এ বিশ্বব্যাপী সম্প্রচারিত এক যুগান্তকারী ঘোষণায়, GrapheneOS Foundation আনুষ্ঠানিকভাবে Motorola Mobility (Lenovo-এর একটি অঙ্গপ্রতিষ্ঠান)-এর সাথে একটি দীর্ঘমেয়াদী কৌশলগত অংশীদারিত্ব উন্মোচন করে। এটি শুধুমাত্র অফ-দ্য-শেলফ কমার্শিয়াল ফোনে OS ইনস্টল করার মতো কোনো সাধারণ বিষয় ছিল না। এটি ছিল গভীর কো-ইঞ্জিনিয়ারিং (co-engineering)-এর একটি প্রক্রিয়া। GrapheneOS-এর কঠোর আবশ্যকতাগুলোকে পূরণ করার জন্য Motorola বিশেষভাবে ডিজাইন করা একটি পারপাস-ইঞ্জিনিয়ারড (purpose-engineered) ডিভাইস তৈরি করেছে।

তাদের উন্নত Snapdragon 8 Elite মাদারবোর্ডগুলোতে কাস্টম hardware কম্পোনেন্টগুলোকে ইন্টিগ্রেট করার মাধ্যমে, Motorola সফলভাবে সেই সব cryptographic সক্ষমতাগুলোকে রেপ্লিকেট করেছে এবং ছাড়িয়ে গেছে, যা আগে শুধুমাত্র Google-এর Titan M চিপের একচেটিয়া অধিকারভুক্ত ছিল। তারা বাধ্যতামূলক Hardware Memory Tagging (MTE), ফুল A/B পার্টিশন সাপোর্ট, এবং hardware-level USB পোর্ট ডিজেবলিং (disabling) ইমপ্লিমেন্ট করেছে।

এই পার্টনারশিপটি সিকিউর কমিউনিকেশনের ক্ষেত্রে একটি জিওপলিটিক্যাল বা ভূ-রাজনৈতিক মাইলফলক। এটিই প্রথমবার, যখন একটি প্রধান Original Equipment Manufacturer (OEM) একটি মাস-মার্কেট (mass-market) এন্টারপ্রাইজ সলিউশন তৈরি করার জন্য একটি স্বাধীন privacy অপারেটিং সিস্টেমের সাথে সরাসরি কোলাবরেট করেছে। Motorola-র hardware-এ প্রসারিত হওয়ার মাধ্যমে, GrapheneOS এটি নিশ্চিত করেছে যে, সাধারণ ব্যবহারকারী এবং এন্টারপ্রাইজ ফ্লিটগুলোকে (enterprise fleets) আর কোনো একক hardware ভেন্ডরের ওপর নির্ভর করতে বাধ্য হতে হবে না। এই একচেটিয়া আধিপত্য ভেঙে দেওয়া হয়েছে এবং স্বাধীন, গাণিতিকভাবে সুরক্ষিত মোবাইল কম্পিউটিংয়ের ভবিষ্যত নিশ্চিত করা হয়েছে।

1.3 Pre-Installation এবং OEM Unlocking

Digital sovereignty বা ডিজিটাল সার্বভৌমত্বের দিকে এই যাত্রাটি শুরু হয় আপনার hardware-কে যথাযথভাবে প্রস্তুত করার মাধ্যমে। আমাদের নতুন, শক্তিশালী বাঙ্কারের জন্য কংক্রিট ঢালাই করার আগে, আমাদেরকে প্রথমেই কনস্ট্রাকশন সাইটটিকে সুরক্ষিত করতে হবে।

অনেক নতুন ব্যবহারকারীর কাছেই একটি custom অপারেটিং সিস্টেম ইনস্টল করার সম্ভাবনাটি অবিশ্বাস্য রকমের ভীতিকর শোনায়। এটি তাদের কল্পনায় জটিল, ম্যাট্রিক্স-স্টাইলের কমান্ড টার্মিনাল এবং একটি ব্যয়বহুল স্মার্টফোনকে "brick" বা পুরোপুরি ধ্বংস করে ফেলার একটি চিরস্থায়ী ভয়ের চিত্র ফুটিয়ে তোলে। একটি দীর্ঘশ্বাস নিন এবং সেই দুশ্চিন্তাটিকে ঝেড়ে ফেলুন।

GrapheneOS Foundation একটি চমৎকার, সম্পূর্ণ স্বয়ংক্রিয় Web Installer তৈরি করেছে যা আপনার হয়ে সবচেয়ে কঠিন কাজগুলো পরিচালনা করে। এটি সম্পন্ন করার জন্য আপনাকে কোনো অভিজ্ঞ সফটওয়্যার ইঞ্জিনিয়ার বা কমান্ড-লাইন উইজার্ড হতে হবে না। আপনি যদি কেক বানানোর রেসিপি অনুসরণ করতে পারেন, তবে আপনি সফলভাবে GrapheneOS-ও ইনস্টল করতে পারবেন।

তবে, এই প্রক্রিয়ার সাফল্য সম্পূর্ণভাবে নির্ভর করে আপনার প্রস্তুতি এবং নির্ভুলতার ওপর। এই ধাপে তাড়াহুড়ো করবেন না। নিচের prerequisite বা পূর্বশর্তগুলোকে কঠোরভাবে মেনে চলার মাধ্যমে, আপনি একটি ত্রুটিমুক্ত এবং স্ট্রেস-ফ্রি ইনস্টলেশন প্রক্রিয়া নিশ্চিত করতে পারবেন।

Prerequisites: প্রয়োজনীয় সরঞ্জাম সংগ্রহ

আপনার নতুন ডিভাইসটি স্পর্শ করার আগেই আপনাকে নিশ্চিত করতে হবে যে আপনার ওয়ার্কস্পেস—বিশেষ করে ইনস্টলেশনটি সম্পন্ন করার জন্য আপনি যে কম্পিউটারটি ব্যবহার করছেন—তা কঠোর অপারেশনাল রিকোয়ারমেন্টগুলো পূরণ করে।

১. পর্যাপ্ত Computing Resources আপনার কম্পিউটারের জন্য ফ্ল্যাশিং প্রসেসটি অত্যন্ত রিসোর্স-ইন্টেনসিভ (resource-intensive)। Web Installer সম্পূর্ণ GrapheneOS অপারেটিং সিস্টেমের একটি হাইলি কমপ্রেসড (highly compressed), cryptographically signed ইমেজ ডাউনলোড করবে। আপনার ফোনে এই ইমেজটি ট্রান্সমিট বা পাঠানোর আগে, এটিকে আপনার কম্পিউটারের মেমরিতে লোকালি আনপ্যাক (unpack) এবং ভেরিফাই করতে হবে। সিস্টেম

ক্র্যাশ ছাড়াই এই কাজটি সম্পন্ন করার জন্য, আপনার ইনস্টলেশন কম্পিউটারে অন্তত **2GB ফ্রি, অ্যাভেইলেবল RAM** এবং ন্যূনতম **32GB ফ্রি স্টোরেজ স্পেস** থাকতে হবে। আপনার কম্পিউটারের হার্ড ড্রাইভ যদি পুরোপুরি পূর্ণ থাকে, তবে প্রক্রিয়ার মাঝামাঝি সময়ে এক্সট্রাকশন (extraction) ফেইল করবে এবং ইনস্টলেশনটি থেমে যাবে।

২. একটি Compatible Web Browser Web Installer মূলত WebUSB নামের একটি স্পেশালাইজড প্রোটোকলের মাধ্যমে আপনার ফোনের hardware-এর সাথে সরাসরি যোগাযোগ করে। সব ব্রাউজার এই প্রোটোকল সাপোর্ট করে না। আপনাকে অবশ্যই একটি আধুনিক, Chromium-ভিত্তিক ব্রাউজার ব্যবহার করতে হবে। এই কাজের জন্য স্ট্যান্ডার্ড Google Chrome, Microsoft Edge, বা Brave Browser হলো রিকমেন্ডেড টুল। Safari, Firefox, বা Tor Browser ব্যবহার করে এই প্রক্রিয়ার চেষ্টা করবেন অ্যাকাউন্ট, কারণ সেগুলোতে প্রয়োজনীয় WebUSB ইন্টিগ্রেশনের ঘাটতি রয়েছে।

৩. ফিজিক্যাল কানেকশন (অত্যন্ত গুরুত্বপূর্ণ) এই পুরো ম্যানুয়ালের মধ্যে এটিই হলো সবচেয়ে গুরুত্বপূর্ণ prerequisite: আপনাকে অবশ্যই একটি হাই-কোয়ালিটি USB-C কেবল ব্যবহার করতে হবে।

তাছাড়া, আপনাকে অবশ্যই কেবলটিকে আপনার কম্পিউটারের প্রাইমারি পোর্টের সাথে **সরাসরি** প্লাগ ইন করতে হবে। আপনি যদি কোনো ডেস্কটপ কম্পিউটার ব্যবহার করেন, তবে টাওয়ারের সামনের প্যানেলে নয়, বরং পেছনের দিকের মাদারবোর্ডে সরাসরি কেবলটিকে প্লাগ করুন। আর যদি ল্যাপটপ ব্যবহার করেন, তবে এটিকে সরাসরি সাইড পোর্টে প্লাগ ইন করুন।

যেকোনো পরিস্থিতিতেই, একটি USB হাব, ডঙ্গল বা মনিটর পাস-থ্রু (pass-through)-এর মাধ্যমে এই কানেকশন রুট করা থেকে বিরত থাকুন। USB হাবগুলো পাওয়ার ফ্লাকচুয়েশন (power fluctuations) এবং ডেটা স্ট্রিমের মাইক্রো-ইন্টারাপশনের (micro-interruptions) জন্য কুখ্যাত। আপনার কম্পিউটার যখন ফোনে নতুন bootloader ফ্ল্যাশ করছে, তখন যদি এক মিলি-সেকেন্ডের ভগ্নাংশের জন্যও ডেটা কানেকশন ড্রপ করে, তবে ইনস্টলেশনটি ভয়ংকরভাবে ফেইল করবে, যা আপনার ডিভাইসটিকে একটি soft-bricked অবস্থায় ফেলে দিতে পারে। তাই একটি বাধাহীন, সরাসরি ফিজিক্যাল কানেকশন থাকা একেবারেই বাধ্যতামূলক।

Carrier-Lock Warning বা ক্যারিয়ার-লক সতর্কবার্তা

ফিজিক্যাল ডিভাইসে অগ্রসর হওয়ার আগে, hardware সোর্সিং সংক্রান্ত একটি অত্যন্ত ক্রিটিক্যাল সতর্কতা বা caveat রয়েছে। GrapheneOS শুধুমাত্র সেইসব ডিভাইসেই ইনস্টল করা যেতে পারে, যেগুলো আপনাকে অপারেটিং সিস্টেম সম্পূর্ণরূপে মুছে ফেলার বা wipe করার অনুমতি দেয়।

আপনি যদি কোনো প্রধান টেলিকমিউনিকেশন ক্যারিয়ারের (যেমন Verizon, AT&T, বা T-Mobile) মাধ্যমে পেমেন্ট প্ল্যানে আপনার স্মার্টফোনটি কিনে থাকেন, তবে আপনার ডিভাইসটি "carrier-locked" হওয়ার প্রবল সম্ভাবনা রয়েছে। ক্যারিয়ারগুলো প্রায়শই ডিভাইস ফার্মওয়্যারকে লকড ডাউন (locked down) করার ম্যান্ডেট দিয়ে থাকে, যাতে আপনি ফোনটিকে অন্য কোনো নেটওয়ার্কে নিয়ে যেতে না পারেন। এর ফলস্বরূপ, ক্যারিয়ার-লকড ডিভাইসগুলো গাণিতিকভাবে বিকল্প অপারেটিং সিস্টেম ইনস্টল করার বিষয়টিকে পুরোপুরি নিষিদ্ধ করে দেয়।

আপনাকে অবশ্যই একটি "factory unlocked" ডিভাইস ব্যবহার করতে হবে। এগুলো সাধারণত সরাসরি Google Store বা Motorola থেকে কেনা হয়, অথবা বড় বড় ইলেকট্রনিক্স রিটেইলারদের কাছ থেকে "unlocked" হিসেবে চিহ্নিত করা থাকে। যদি আপনি নিচের ধাপগুলো চেষ্টা করেন এবং দেখেন যে আপনি এগোতে পারছেন না, তবে আপনাকে আপনার ক্যারিয়ারের সাথে যোগাযোগ করে ডিভাইসটিকে সম্পূর্ণ আনলক করার জন্য অনুরোধ করতে হবে।

ধাপে ধাপে: OEM Unlocking এনাবেল করা

আপনার কম্পিউটার এবং সরাসরি সংযুক্ত USB কেবলটি প্রস্তুত হওয়ার পর, এবার স্মার্টফোনটির দিকে মনোযোগ দেওয়ার সময় এসেছে। প্রথম লক্ষ্যটি হলো ফিজিক্যাল hardware-এর সাথে একটি ডিজিটাল হ্যান্ডশেক (digital handshake) এক্সিকিউট করা, যা ডিভাইসটিকে সিগন্যাল দেবে যে আপনিই এর প্রকৃত মালিক এবং আপনি এর কোর (core) প্রোগ্রামিং পরিবর্তন করার অধিকার দাবি করছেন। এই প্রক্রিয়াটিকেই মূলত "OEM Unlocking" এনাবেল করা বলা হয়।

নিচের ধাপগুলো ঠিক যেভাবে লেখা আছে, সেভাবেই অনুসরণ করুন:

ধাপ ১: Initial Boot আপনার সম্পূর্ণ নতুন এবং factory-unlocked ডিভাইসটি চালু করুন। ডিভাইসটিতে স্ট্যান্ডার্ড Android সেটআপ স্ক্রিনটি আপনাকে স্বাগত জানাবে। এখানে আপনার মূল লক্ষ্য হলো মানবিকভাবে যত দ্রুত সম্ভব এই সেটআপটি স্কিপ (skip) করে যাওয়া:

- কোনো PIN বা পাসওয়ার্ড সেট করবেন না।
- কোনো ডায়াগনস্টিক ডেটা (diagnostic data) পাঠানোর বিষয়ে সম্মতি জানাবেন না।
- সবচেয়ে গুরুত্বপূর্ণ বিষয় হলো, **কোনো Google অ্যাকাউন্টে সাইন ইন করবেন না** অথবা কোনো ব্যাকআপ রিস্টোর করার চেষ্টা করবেন না।

আপনি এই ডিভাইসটিকে সম্পূর্ণরূপে wipe করতে যাচ্ছেন, তাই আপনি এখন এতে যে ডেটাই দিন না কেন, তা প্রায় সাথে সাথেই ধ্বংস হয়ে যাবে।

ধাপ ২: Wi-Fi-এর সাথে কানেক্ট করা সেটআপ স্কিপ করার ক্ষেত্রে শুধুমাত্র একটি ব্যতিক্রম রয়েছে: আপনাকে অবশ্যই Wi-Fi-এর মাধ্যমে ডিভাইসটিকে ইন্টারনেটের সাথে কানেক্ট করতে হবে। এটি একটি অত্যন্ত ক্রিটিক্যাল এবং প্রায়শই উপেক্ষিত ধাপ। আধুনিক hardware-এ bootloader খোলার অনুমতি দেওয়ার আগে, ম্যানুফ্যাকচারারের সার্ভারে একটি দ্রুত পিং (ping) করতে হয়। এটি নিশ্চিত করে যে ডিভাইসটির সিরিয়াল নম্বরটি কোনোভাবেই ব্ল্যাকলিস্টেড (blacklisted) বা স্থায়ীভাবে ক্যারিয়ার-লকড (carrier-locked) নয়।

ধাপ ৩: Settings অ্যাক্সেস করা একবার স্ট্যান্ডার্ড Android হোম স্ক্রিনে পৌঁছে গেলে, অ্যাপ ড্রয়ার (app drawer) ওপেন করতে উপরের দিকে সোয়াইপ করুন এবং `Settings` আইকনে (গিয়ার সিম্বল) ট্যাপ করুন।

ধাপ ৪: Build Number খুঁজে বের করা Settings মেনুর একদম নিচের দিকে স্ক্রোল করুন এবং `About phone`-এ ট্যাপ করুন। এই মেনুর ভেতরে, একেবারে নিচে স্ক্রোল করতে থাকুন যতক্ষণ না আপনি `Build number` লেবেলযুক্ত একটি টেক্সটের লাইন দেখতে পান।

ধাপ ৫: Developer Handshake এবার আপনি একটি লুকানো ইঞ্জিনিয়ারিং মেনু আনলক করতে যাচ্ছেন। `Build number`-এর ওপর দ্রুত **ঠিক ৭ বার** ট্যাপ করুন। আপনি যখন ট্যাপ করবেন, তখন একটি ছোট্ট নোটিফিকেশন আসবে যা আপনার ট্যাপগুলো কাউন্টডাউন করবে। সপ্তম ট্যাপে, স্ক্রিনে একটি মেসেজ ফ্ল্যাশ করবে: *"You are now a developer!"*

ধাপ ৬: Developer Options-এ প্রবেশ করা মূল Settings মেনুতে ফিরে যাওয়ার জন্য স্ক্রিনের উপরের বাঁ দিকের ব্যাক অ্যারোটিতে (back arrow) প্রেস করুন। `System`-এ স্ক্রোল করুন এবং এটিতে ট্যাপ করুন। এই মেনুর একেবারে নিচে, আপনি এখন `Developer options` নামে একটি সম্পূর্ণ নতুন এবং আগে লুকিয়ে থাকা অপশন দেখতে পাবেন। এটিতে ট্যাপ করে প্রবেশ করুন।

ধাপ ৭: OEM Unlocking টগল করা Developer options মেনুর ভেতরে সামান্য নিচে স্ক্রোল করুন। আপনি **OEM unlocking** শিরোনামের একটি নির্দিষ্ট সেটিং খুঁজছেন। এটিকে চালু করার জন্য এর পাশের টগল সুইচটিতে ট্যাপ করুন। আপনি OEM unlocking এনাবেল করতে চান কি না, তা জিজ্ঞাসা করে ফোনটি একটি প্রম্পট (prompt) প্রদর্শন করবে। `Enable`-এ ট্যাপ করুন।

(নোট: যদি OEM unlocking টগলটি গ্রে আউট (grayed out) হয়ে থাকে এবং ট্যাপ করা না যায়, তবে ধাপ ২-এ ফিরে যান এবং নিশ্চিত করুন যে আপনি ইন্টারনেটের সাথে সংযুক্ত আছেন। আপনি যদি Wi-Fi-এর সাথে কানেক্টেড থাকেন এবং কয়েক মিনিট পরও টগলটি স্থায়ীভাবে গ্রে আউট হয়ে থাকে, তবে দুর্ভাগ্যবশত আপনার ডিভাইসটি ক্যারিয়ার-লকড (carrier-locked) অবস্থায় রয়েছে।)

আপনার ডিভাইসটি এখন প্রাইমড (primed) এবং একটি নতুন অপারেটিং সিস্টেম গ্রহণ করার জন্য গাণিতিকভাবে প্রস্তুত। আপনি সফলভাবে প্রথম ডিফেন্সিভ দেয়ালটিকে নামিয়ে এনেছেন এবং hardware-কে নির্দেশ দিয়েছেন যেন এটি আপনাকে এর সুপ্রিম অ্যাডমিনিস্ট্রেটর (supreme administrator) হিসেবে স্বীকৃতি দেয়। পরবর্তী সেকশনে, আমরা আপনার কম্পিউটারের সাথে কানেকশন স্থাপন করব এবং আনুষ্ঠানিকভাবে আমাদের ফাউন্ডেশন তৈরি করা শুরু করব। আপনি দারুণ করছেন—চলুন সামনে এগিয়ে যাওয়া যাক।

1.4 The Web Installer Method

আপনার ডিভাইস প্রস্তুত। আপনার কম্পিউটার রেডি। USB কেবলটি সরাসরি পোর্টে প্লাগ ইন করা আছে। একটি ক্রিপ্টোগ্রাফিক ফোর্ট্রেস (cryptographic fortress) দিয়ে সার্ভিল্যান্স ইঞ্জিনকে প্রতিস্থাপন করার সময় এখনই।

GrapheneOS Foundation অপারেটিং সিস্টেমটি ইনস্টল করার জন্য দুটি অফিশিয়াল পদ্ধতি প্রদান করে: অ্যাডভান্সড ব্যবহারকারীদের জন্য একটি command-line interface (CLI), এবং অন্যান্য সকলের জন্য একটি WebUSB-ভিত্তিক Web Installer। আমরা Web Installer ব্যবহার করব। এটি রিকমেন্ডেড বা প্রস্তাবিত পদ্ধতি, যাকে এমনভাবে ইঞ্জিনিয়ারিং করা হয়েছে যেন এত বড় মাত্রার একটি প্রক্রিয়া যতটা সম্ভব ফুলপ্রুফ (foolproof) হতে পারে। এই ইনস্টলারটি আপনার ওয়েব ব্রাউজারের মাধ্যমে সরাসরি আপনার ফোনের hardware-level bootloader-এর সাথে যোগাযোগ করে। এটি firmware ফ্ল্যাশিং, পার্টিশন রাইটিং (partition writing) এবং ক্রিপ্টোগ্রাফিক key এনরোলমেন্টের মতো জটিল কাজগুলোকে স্বয়ংক্রিয় করে দেয়, যার জন্য অন্যথায় ডজন ডজন ম্যানুয়াল টার্মিনাল কমান্ড দেওয়ার প্রয়োজন হতো।

পুরো প্রক্রিয়াটি পাঁচটি ভিন্ন ভিন্ন ধাপ অনুসরণ করে: **Fastboot-এ বুট করা → Bootloader আনলক করা → Release ফ্ল্যাশ করা → Bootloader লক করা → GrapheneOS-এ বুট করা।** প্রতিটি ধাপ অত্যন্ত নির্ভুলভাবে অনুসরণ করুন। কোনো কিছু নিজের মতো করে ইম্প্রোভাইজ (improvise) করার চেষ্টা করবেন না।

> ***Warning: Total Data Destruction*** *ইনস্টলেশন প্রক্রিয়াটি আপনার ডিভাইসের* ***সমস্ত ডেটা সম্পূর্ণ এবং অপরিবর্তনীয়ভাবে wipe বা মুছে ফেলবে****—তাও একবার নয়, দুইবার। bootloader আনলক করলে তা প্রথমবার একটি সম্পূর্ণ factory reset ট্রিগার করে। আর শেষে bootloader লক করলে তা দ্বিতীয়বারের মতো একটি সম্পূর্ণ factory reset ট্রিগার করে। এটি থেকে ডেটা রিকভার বা পুনরুদ্ধার করার কোনো সুযোগ নেই। যদি আপনার এই ফোনে এমন কোনো ডেটা থেকে থাকে যা আপনার প্রয়োজন, তবে সামনে এগোনোর আগেই তা একটি এক্সটার্নাল ড্রাইভে ব্যাকআপ করে রাখুন। একবার আমরা শুরু করলে, এই ডিভাইসের সবকিছু গাণিতিকভাবে ধ্বংস হয়ে যাবে।*

Browser Compatibility: শুরু করার আগে

Web Installer আপনার ফোনের fastboot ইন্টারফেসের সাথে যোগাযোগ করতে WebUSB প্রোটোকলের ওপর নির্ভর করে। এই প্রোটোকলটি শুধুমাত্র নির্দিষ্ট কিছু ব্রাউজার সাপোর্ট করে। ইনস্টলারে নেভিগেট করার আগে, নিশ্চিত হয়ে নিন যে আপনি নিচের যেকোনো একটি ব্যবহার করছেন:

- **Google Chrome** — সম্পূর্ণরূপে সাপোর্টেড। সবচেয়ে নির্ভরযোগ্য অপশন।
- **Microsoft Edge** — সম্পূর্ণরূপে সাপোর্টেড (Chromium-ভিত্তিক)।
- **Brave Browser** — সাপোর্টেড, তবে শুরু করার আগে আপনাকে **অবশ্যই Brave Shields ডিজেবল** করতে হবে। Brave-এর অ্যান্টি-ফিঙ্গারপ্রিন্টিং প্রোটেকশন লোকাল স্টোরেজ ব্যবহারকে একটি নির্দিষ্ট নিম্ন মাত্রায় সীমাবদ্ধ করে রাখে, যা ডাউনলোড করা OS ইমেজ এক্সট্র্যাক্ট করার জন্য ইনস্টলারকে পর্যাপ্ত স্পেস পেতে বাধা দেবে। `brave://settings/shields`-এ নেভিগেট করুন এবং ইনস্টলার পেইজটির জন্য সাময়িকভাবে Shields ডিজেবল করে দিন।
- **Vanadium** — GrapheneOS-এর নিজস্ব হার্ডেনড ব্রাউজার। যদি আপনি অন্য কোনো GrapheneOS ডিভাইস থেকে ইনস্টল করে থাকেন, তবে এটি সম্পূর্ণরূপে সাপোর্টেড।
- **Chromium** — বেশিরভাগ প্ল্যাটফর্মে সাপোর্টেড, তবে Ubuntu Snap প্যাকেজ ভার্সনটি এড়িয়ে চলুন, কারণ এটি ব্রোকেন (broken) WebUSB-এর সাথে আসে।

নিচেরগুলো ব্যবহার করবেন না:

- **Firefox** — WebUSB সাপোর্ট করে না। ইনস্টলার আপনার ডিভাইসটি ডিটেক্ট করতে পারবে না।
- **Safari** — WebUSB সাপোর্ট করে না।
- **Tor Browser** — WebUSB সাপোর্ট করে না।
- **Incognito বা Private Browsing মোডে থাকা যেকোনো ব্রাউজার** — Private মোডগুলো ওয়েব অ্যাপ্লিকেশনের জন্য লোকাল স্টোরেজ কোটাকে মারাত্মকভাবে সীমিত করে দেয়। ইনস্টলারকে টেম্পোরারি ব্রাউজার স্টোরেজে একটি মাল্টি-গিগাবাইট OS ইমেজ ডাউনলোড এবং এক্সট্র্যাক্ট করতে হয়। Incognito মোড পর্যাপ্ত স্পেস অ্যালোকেট (allocate) করবে না এবং এক্সট্র্যাকশন প্রক্রিয়াটি নীরবে ফেইল করবে।
- **ব্রাউজারের Flatpak বা Snap ভার্সন** — এই স্যান্ডবক্সড (sandboxed) প্যাকেজ ফরম্যাটগুলো ফ্ল্যাশিং প্রক্রিয়ার সময় USB কানেক্টিভিটি ইস্যু তৈরি করার জন্য পরিচিত।

ফেজ ১: Fastboot Mode-এ বুট করা

প্রথম ধাপটি হলো আপনার ফোনটিকে এর লো-লেভেল (low-level) bootloader ইন্টারফেসে প্রবেশ করানো, যা **Fastboot Mode** নামে পরিচিত। এই অবস্থায় ফোনের অপারেটিং সিস্টেম রান করে না। ডিভাইসটি একটি প্রি-বুট (pre-boot) পরিবেশে অবস্থান করে এবং আপনার কম্পিউটার থেকে র (raw) hardware-level কমান্ডের জন্য অপেক্ষা করতে থাকে।

1. ডিভাইসটিকে সম্পূর্ণভাবে **Power off** বা বন্ধ করুন।

2. ফোনের পাশে থাকা **Volume Down** বাটনটি চেপে ধরে রাখুন।
3. Volume Down বাটনটি চেপে ধরা অবস্থাতেই, ডিভাইসটি চালু করার জন্য **Power** বাটনটি চেপে ধরুন।
4. দুটি বাটনই চেপে ধরে রাখুন। কয়েক সেকেন্ড পর, স্বাভাবিক বুট অ্যানিমেশনের পরিবর্তে, আপনি একটি মিনিমাল স্ক্রিন দেখতে পাবেন যেখানে একটি লাল সতর্কীকরণ ত্রিভুজ এবং "**Fastboot Mode**" টেক্সটটি প্রদর্শিত হবে।
5. **ফোনে আর কিছুই প্রেস করবেন না।** কোনো মেনু আইটেম সিলেক্ট করবেন না। Web Installer-এর সাথে সংযুক্ত হওয়ার জন্য ডিভাইসটিকে অবশ্যই Fastboot Mode-এ পজ বা থামানো অবস্থায় থাকতে হবে। এটিকে এভাবেই এই স্ক্রিনে রেখে দিন।

ফেজ ২: ডিভাইস কানেক্ট করা এবং Bootloader আনলক করা

Fastboot Mode স্ক্রিন প্রদর্শিত থাকা অবস্থায়, এখন ফিজিক্যাল লিংক বা সংযোগ স্থাপন করার এবং ভল্টের দরজা খোলার সময় এসেছে।

1. আপনার কম্পিউটারে compatible ব্রাউজারটি ওপেন করুন।
2. অফিশিয়াল GrapheneOS Web Installer-এ নেভিগেট করুন: `https://grapheneos.org/install/web`
3. USB-C কেবল ব্যবহার করে আপনার ফোনটিকে কম্পিউটারের সাথে কানেক্ট করুন। মনে রাখবেন: সরাসরি কম্পিউটারের পোর্টে প্লাগ ইন করুন। কোনো হাব নয়, কোনো ডঙ্গল নয়।
4. Web Installer পেইজে, "**Unlock Bootloader**" লেবেলযুক্ত বাটনটি খুঁজে বের করুন এবং ক্লিক করুন।
5. আপনার ব্রাউজারে একটি পপ-আপ উইন্ডো প্রদর্শিত হবে, যেখানে আপনাকে একটি USB ডিভাইস সিলেক্ট করতে বলা হবে। তালিকা থেকে আপনার Pixel ডিভাইসটি সিলেক্ট করুন এবং "**Connect**"-এ ক্লিক করুন।
6. **আপনার ফোনের স্ক্রিনের দিকে লক্ষ্য করুন।** ডিভাইসটিতে এখন একটি কনফারমেশন প্রম্পট দেখা যাবে, যেখানে জানতে চাওয়া হবে আপনি bootloader আনলক করতে চান কি না। এটি আপনাকে সতর্ক করে দেবে যে সমস্ত ডেটা মুছে ফেলা হবে। ফোনে থাকা **Volume buttons** ব্যবহার করে অন-স্ক্রিন সিলেকশনে নেভিগেট করে "**Unlock the bootloader**" অপশনটিতে যান। এরপর কনফার্ম করার জন্য **Power button** প্রেস করুন।
7. ফোনটি একটি factory reset সম্পন্ন করবে। এটি হলো প্রথম ডেটা wipe। কিছুক্ষণ পর, ডিভাইসটি পুনরায় রিবুট হয়ে Fastboot Mode-এ ফিরে আসবে।

ফেজ ৩: GrapheneOS Release ফ্ল্যাশ করা

bootloader এখন উন্মুক্ত। আপনার ফোনটি এখন একটি খালি ভল্ট, যা এর নতুন অপারেটিং সিস্টেম গ্রহণ করার জন্য পুরোপুরি প্রস্তুত।

1. ব্রাউজারে পুনরায় Web Installer পেইজে ফিরে যান এবং "**Download Release**" নামক সেকশনটি খুঁজুন। ডাউনলোড বাটনে ক্লিক করুন। ইনস্টলার স্বয়ংক্রিয়ভাবে আপনার নির্দিষ্ট Pixel মডেলটি ডিটেক্ট করবে এবং আপনার ডিভাইসের জন্য সঠিক ও ক্রিপ্টোগ্রাফিক্যালি সাইনড (cryptographically signed) factory image ডাউনলোড করা শুরু করবে। এই ফাইলটি কয়েক গিগাবাইটের হয়ে থাকে; ডাউনলোডের সময় আপনার ইন্টারনেটের গতির ওপর নির্ভর করবে।
2. ডাউনলোড সম্পন্ন হওয়ার পর, ইনস্টলার স্বয়ংক্রিয়ভাবে ইমেজটির ক্রিপ্টোগ্রাফিক সিগনেচার ভেরিফাই করবে, যাতে নিশ্চিত হওয়া যায় যে ডাউনলোডের সময় এর সাথে কোনো টেম্পারিং (tampering) করা হয়নি।
3. "**Flash Release**" লেবেলযুক্ত বাটনে ক্লিক করুন।
4. ইনস্টলার এবার মাল্টি-স্টেজ ফ্ল্যাশিং প্রক্রিয়া শুরু করবে। এটি প্রথমে আপনার ডিভাইসে লেটেস্ট firmware ফ্ল্যাশ করবে, এরপর ফোনটিকে রিবুট করে পুনরায় Fastboot Mode-এ নিয়ে যাবে, এবং সবশেষে GrapheneOS অপারেটিং সিস্টেম পার্টিশনগুলো ফ্ল্যাশ করবে। **ফোনটি স্পর্শ করবেন না। USB কেবলটি ডিসকানেক্ট করবেন না। কোনোভাবেই ডিভাইসটির সাথে ইন্টার‍্যাক্ট করবেন না।** এই প্রক্রিয়ায় একাধিকবার স্বয়ংক্রিয় রিবুট হয়ে থাকে এবং যেকোনো ধরনের বাধা ডিভাইসটিকে একটি অসম্পূর্ণ অবস্থায় ফেলে দিতে পারে।
5. রিবুট হওয়ার পর যদি আপনার ব্রাউজার আপনাকে USB ডিভাইসের সাথে পুনরায় কানেক্ট করার প্রম্পট দেখায় (এটি একটি সাধারণ বিষয়), তবে প্রম্পট অনুযায়ী "**Reconnect Device**"-এ ক্লিক করুন। ইনস্টলার খুব সুন্দরভাবেই এটি পরিচালনা করে।
6. ধৈর্য ধরে অপেক্ষা করুন। আপনার hardware-এর ওপর ভিত্তি করে ফ্ল্যাশিং প্রক্রিয়াটি সম্পন্ন হতে সাধারণত ৫ থেকে ১৫ মিনিট সময় লাগে।

ফেজ ৪: Bootloader লক করা — সবচেয়ে ক্রিটিক্যাল ধাপ

অপারেটিং সিস্টেমটি ফ্ল্যাশ করা হয়েছে। GrapheneOS এখন আপনার ডিভাইসের স্টোরেজ পার্টিশনগুলোতে ফিজিক্যালি লেখা বা রাইট করা হয়ে গেছে। তবে, আপনার ফোনটি এখনও সুরক্ষিত নয়। এই মুহূর্তে, bootloader এখনও আনলক করা অবস্থায় আছে, যার মানে হলো আপনার ডিভাইসের ফিজিক্যাল অ্যাক্সেস আছে এমন যেকেউ আপনার OS-এর ওপর একটি ম্যালিশিয়াস (malicious) OS ফ্ল্যাশ করে দিতে পারে। এটিই পুরো ইনস্টলেশন প্রক্রিয়ার সবচেয়ে গুরুত্বপূর্ণ ধাপ।

1. ফ্ল্যাশিং প্রক্রিয়া সম্পন্ন হওয়ার পর, আপনার ডিভাইসটি পুনরায় Fastboot Mode-এ ফিরে আসবে। Web Installer পেইজে, "**Lock Bootloader**" লেবেলযুক্ত বাটনটি খুঁজে বের করুন এবং ক্লিক করুন।
2. **আপনার ফোনের দিকে লক্ষ্য করুন।** ডিভাইসটি আবার একটি কনফারমেশন প্রম্পট প্রদর্শন করবে। **Volume buttons** ব্যবহার করে, "**Lock the bootloader**" অপশনটিতে নেভিগেট করুন এবং কনফার্ম করার জন্য **Power button** প্রেস করুন।

3. ফোনটি দ্বিতীয়বারের মতো একটি সম্পূর্ণ factory reset সম্পন্ন করবে। এটি প্রত্যাশিত এবং ডিজাইনেরই একটি অংশ।

এই ধাপটি এত ক্রিটিক্যাল কেন? bootloader লক করার মাধ্যমে GrapheneOS-এর নিজস্ব ক্রিপ্টোগ্রাফিক সাইনিং key দিয়ে **Android Verified Boot** অ্যাক্টিভেট হয়। এই মুহূর্তের পর থেকে, প্রতিবার ফোন চালু হওয়ার সময়, ডিভাইসের ফিজিক্যাল silicon গাণিতিকভাবে ভেরিফাই করবে যে সম্পূর্ণ অপারেটিং সিস্টেমটি—প্রতিটি পার্টিশন, প্রতিটি firmware blob, কোডের প্রতিটি লাইন—GrapheneOS-এর একটি অথেনটিক এবং আনটেম্পারড (untampered) বিল্ড। এমনকি যদি এক বাইট ডেটাও অসৎ উদ্দেশ্যে পরিবর্তন করা হয়ে থাকে, তবে ডিভাইসটি বুট হতে পুরোপুরি অস্বীকার করবে। এটি গাণিতিক গ্যারান্টি প্রদান করে যে কোনো "evil maid" অ্যাটাক, কোনো স্টেট-স্পন্সরড ইমপ্ল্যান্ট (state-sponsored implant), এবং কোনো সাপ্লাই-চেইন কম্প্রোমাইজ (supply-chain compromise) নীরবে আপনার অপারেটিং সিস্টেমে কোড ইনজেক্ট করতে পারবে না। bootloader লক করা ছাড়া আপনার কাছে কেবল একটি de-Googled ফোন রয়েছে। কিন্তু এটি লক করার মাধ্যমেই আপনি একটি সত্যিকারের ক্রিপ্টোগ্রাফিক ফোর্ট্রেস (cryptographic fortress) অর্জন করবেন।

ফেজ ৫: প্রথমবার বুট করা

1. bootloader লক হয়ে যাওয়ার পর, ফোনটি পুনরায় Fastboot Mode স্ক্রিনে ফিরে আসবে। ডিফল্ট অপশন হিসেবে "**Start**" হাইলাইট করা থাকার কথা। বুট করার জন্য **Power button** প্রেস করুন।
2. ডিভাইসটি পাওয়ার অন হবে। আপনি বুট স্ক্রিনে একটি সংক্ষিপ্ত হলুদ নোটিশ দেখতে পাবেন যা নির্দেশ করে যে একটি বিকল্প বা অল্টারনেট OS লোড করা হয়েছে—লক করা bootloader-সহ যেকোনো নন-স্টক অপারেটিং সিস্টেমের জন্য এটি সম্পূর্ণ স্বাভাবিক এবং প্রত্যাশিত। Pixel 6 এবং এর পরবর্তী ডিভাইসগুলোতে সম্পূর্ণ verified boot key hash প্রদর্শিত হয়, যা আপনি চূড়ান্ত অথেনটিসিটি (authenticity) নিশ্চিত করার জন্য GrapheneOS-এর ওয়েবসাইটে প্রকাশিত অফিশিয়াল হ্যাশগুলোর সাথে ক্রস-রেফারেন্স করে মিলিয়ে দেখতে পারেন।
3. কিছুক্ষণ পর, GrapheneOS সেটআপ উইজার্ড আপনাকে স্বাগত জানাবে।
4. সেটআপের একদম শেষের স্ক্রিনে, আপনি **disable OEM unlocking**-এর একটি টগল দেখতে পাবেন। এই টগলটি ডিফল্টভাবেই চেক করা থাকে। **এটিকে চেক করাই থাকতে দিন।** OEM unlocking ডিজেবল করার মাধ্যমে এটি নিশ্চিত করা হয় যে, সম্পূর্ণ অথেনটিকেটেড (authenticated) অপারেটিং সিস্টেমে প্রথমে বুট না করে (যার জন্য আপনার পাসকোড প্রয়োজন) কেউই bootloader-এ পুনরায় প্রবেশ করতে এবং ডিভাইসের সাথে টেম্পারিং করতে পারবে না।

অভিনন্দন। আপনি এখন এমন একটি ডিভাইস ধরে আছেন যা আপনি ছাড়া আর কোনো মাস্টারের সেবা করে না। সার্ভিল্যান্স ইঞ্জিনটিকে ধ্বংস করে দেওয়া হয়েছে। ক্রিপ্টোগ্রাফিক key-গুলো এনরোল করা হয়েছে। Verified Boot এখন পুরোপুরি অ্যাক্টিভ।

আপনার ফাউন্ডেশন ঢালাই করা সম্পন্ন হয়েছে, এবং এটি তৈরি হয়েছে রেইনফোর্সড, গাণিতিকভাবে ভেরিফাই করা কংক্রিট দিয়ে। পরবর্তী চ্যাপ্টারে, আমরা আপনার দৈনন্দিন অপারেশনাল ব্যবহারের জন্য এই ফোর্ট্রেসটিকে কনফিগার করা শুরু করব।

১.৫ কমান্ড লাইন ইন্টারফেস (CLI) ইনস্টলেশন

পূর্ববর্তী সেকশনে বর্ণিত Web Installer হলো অধিকাংশ ব্যবহারকারীর জন্য প্রস্তাবিত মেথড। তবে, সিস্টেম অ্যাডমিনিস্ট্রেটর, পেনিট্রেশন টেস্টার এবং অ্যাডভান্সড Linux অপারেটর যারা ফ্ল্যাশিং প্রসেসের প্রতিটি ধাপে গ্র্যানুলার টার্মিনাল কন্ট্রোল পছন্দ করেন, তাদের জন্য GrapheneOS একটি সম্পূর্ণ ডকুমেন্টেড কমান্ড লাইন ইন্টারফেস (CLI) ইনস্টলেশন পাথ প্রদান করে।

এই CLI মেথড আপনাকে `fastboot` প্রোটোকলে সরাসরি এবং আনমিডিয়েটেড অ্যাক্সেস দেয়। এখানে প্রতিটি কমান্ড এক্সপ্লিসিট। প্রতিটি অপারেশন দৃশ্যমান। আপনার এবং ডিভাইসের র (raw) সিলিকনের মধ্যে কোনো ব্রাউজার অ্যাবস্ট্রাকশন, কোনো WebUSB হ্যান্ডশেক, এবং কোনো গ্রাফিক্যাল ইন্টারফেস থাকে না। আপনি যদি টার্মিনাল নেভিগেট করতে স্বাচ্ছন্দ্যবোধ করেন, তবে এই মেথড সর্বোচ্চ ট্রান্সপারেন্সি এবং কন্ট্রোল অফার করে।

> ***Warning*** *Web Installer-এর মতোই, CLI প্রসেস টার্গেট ডিভাইসের* ***সমস্ত ডেটা সম্পূর্ণ এবং ইরিভার্সিবলি ওয়াইপ (wipe) করে দেবে*** *— তা-ও দুইবার। bootloader আনলক করলে প্রথম ফ্যাক্টরি রিসেট ট্রিগার হয়, এবং bootloader লক করলে দ্বিতীয় ফ্যাক্টরি রিসেট ট্রিগার হয়। সামনে অগ্রসর হওয়ার আগে যেকোনো ক্রিটিক্যাল ডেটার ব্যাকআপ নিয়ে রাখুন।*

ধাপ ১: ডিপেন্ডেন্সি ইনস্টল করা

CLI ইনস্টলেশন দুটি কোর ইউটিলিটির ওপর নির্ভর করে: `fastboot` (ডিভাইসের bootloader-এর সাথে কমিউনিকেট করার জন্য) এবং `openssh` (ডাউনলোডকৃত ফ্যাক্টরি ইমেজগুলোকে ক্রিপ্টোগ্রাফিক্যালি ভেরিফাই করার জন্য)। GrapheneOS রিলিজ সার্ভার থেকে ফাইল ডাউনলোড করার জন্য আপনার সিস্টেমে অবশ্যই `curl` থাকতে হবে।

Arch Linux-এ:

Arch একটি ওয়েল-মেইনটেইনড প্যাকেজ প্রদান করে। একটি সিঙ্গেল কমান্ডে `android-tools` (যার মধ্যে `fastboot` অন্তর্ভুক্ত) এবং `openssh` ইনস্টল করুন:

```
sudo pacman -S android-tools openssh
```

পাশাপাশি, আপনার ইউজার অ্যাকাউন্টকে root প্রিভিলেজ ছাড়াই USB ডিভাইসের সাথে কমিউনিকেট করতে পারমিশন দেওয়ার জন্য `android-udev` প্যাকেজটি ইনস্টল করুন:

```
sudo pacman -S android-udev
```

Debian এবং Ubuntu-তে:

Debian/Ubuntu রিপোজিটরিগুলো `fastboot`-এর ব্রোকেন এবং অত্যন্ত আউটডেটেড ভার্সন শিপ করে। সেগুলো **কখনোই** ব্যবহার করবেন পুনরায়। এর পরিবর্তে, সরাসরি Google থেকে অফিসিয়াল স্ট্যান্ডঅ্যালোন `platform-tools` প্যাকেজ ডাউনলোড করুন। রিপোজিটরি থেকে `openssh-client` এবং udev রুলস ইনস্টল করুন:

```
sudo apt install openssh-client android-sdk-platform-tools-common libarchive-
tools
```

এরপর স্ট্যান্ডঅ্যালোন platform-tools ডাউনলোড এবং ভেরিফাই করুন:

```
curl -O https://dl.google.com/android/repository/platform-tools_r35.0.2-
linux.zip
echo 'acfdcccb123a8718c46c46c059b2f621140194e5ec1ac9d81715be3d6ab6cd0a
platform-tools_r35.0.2-linux.zip' | sha256sum -c
bsdtar xvf platform-tools_r35.0.2-linux.zip
```

macOS-এ:

macOS-এ `openssh` প্রি-ইনস্টল করা থাকে। macOS-এর platform-tools ডাউনলোড এবং ভেরিফাই করুন:

```
curl -O https://dl.google.com/android/repository/platform-tools_r35.0.2-
darwin.zip
echo 'SHA256 (platform-tools_r35.0.2-darwin.zip) =
1820078db90bf21628d257ff052528af1c61bb48f754b3555648f5652fa35d78' | shasum -c
tar xvf platform-tools_r35.0.2-darwin.zip
```

Windows-এ:

একটি **রেগুলার (নন-অ্যাডমিনিস্ট্রেটর) PowerShell** টার্মিনাল ওপেন করুন। লিগ্যাসি Command Prompt ব্যবহার করবেন না। প্রথমে, PowerShell-এর বিল্ট-ইন `curl` অ্যালিয়াসটি রিমুভ করুন যাতে এটি আসল `curl.exe`-এর সাথে কনফ্লিক্ট না করে:

```
Remove-Item Alias:Curl
```

এরপর Windows-এর platform-tools ডাউনলোড এবং ভেরিফাই করুন:

```
curl -O https://dl.google.com/android/repository/platform-tools_r35.0.2-win.zip
(Get-FileHash platform-tools_r35.0.2-win.zip).hash -eq
"2975a3eac0b19182748d64195375ad056986561d994fffbdc64332a516300bb9"
tar xvf platform-tools_r35.0.2-win.zip
```

ধাপ ২: আপনার PATH-এ `fastboot` যুক্ত করা

ফ্ল্যাশিং স্ক্রিপ্ট চালানোর জন্য টার্মিনালের PATH থেকে `fastboot` অ্যাক্সেসযোগ্য হওয়া প্রয়োজন। এটি একটি অত্যন্ত গুরুত্বপূর্ণ ধাপ — এটি ছাড়া ইনস্টলেশন ফেইল করবে।

Linux এবং macOS-এ:

```
export PATH="$PWD/platform-tools:$PATH"
```

Windows (PowerShell)-এ:

```
$env:Path = "$pwd\platform-tools;$env:Path"
```

ভার্সন চেক করে ইনস্টলেশন ভেরিফাই করুন। এটি অবশ্যই **ন্যূনতম 35.0.1** হতে হবে:

```
fastboot --version
```

আপনি নিচের মতো অনুরূপ একটি আউটপুট দেখতে পাবেন:

```
fastboot version 35.0.2-12147458
Installed as /home/username/platform-tools/fastboot
```

ধাপ ৩: Fastboot Mode-এ বুট করা এবং Bootloader আনলক করা

ডিপেন্ডেন্সি ইনস্টল করা হলে এবং ডিভাইসে OEM Unlocking এনাবেল করার পর (যা সেকশন ১.৩-এ আলোচনা করা হয়েছে), ফোনটি পাওয়ার অফ করুন এবং bootloader ইন্টারফেসে বুট করুন:

1. **Volume Down** বাটনটি চেপে ধরে রাখুন।
2. বাটনটি ধরে রাখা অবস্থায়, **Power** বাটন প্রেস করুন।
3. ডিভাইসটি একটি মিনিমাল স্ক্রিনে বুট হবে, যেখানে একটি লাল ওয়ার্নিং ট্রায়াঙ্গেল এবং "**Fastboot Mode**" টেক্সটটি দেখাবে। এই অবস্থায় অন্য কোনো বাটন প্রেস করবেন না।

একটি USB-C ক্যাবল দিয়ে ডিভাইসটিকে আপনার কম্পিউটারের সাথে কানেক্ট করুন (সরাসরি পোর্টে — কোনো হাব ব্যবহার করবেন না)। এরপর, bootloader আনলক করুন:

```
fastboot flashing unlock
```

ডিভাইসে একটি কনফার্মেশন প্রম্পট প্রদর্শিত হবে। **Volume বাটনগুলো** ব্যবহার করে "Unlock the bootloader"-এ নেভিগেট করুন এবং কনফার্ম করার জন্য **Power বাটন** প্রেস করুন। ডিভাইসটি একটি কমপ্লিট ফ্যাক্টরি রিসেট পারফর্ম করবে এবং পুনরায় Fastboot Mode-এ রিবুট হবে।

ধাপ ৪: ফ্যাক্টরি ইমেজ ডাউনলোড এবং ভেরিফাই করা

এখান থেকেই CLI মেথড একটি ডিসাইসিভ সিকিউরিটি অ্যাডভান্টেজ প্রদান করে। আপনি স্বাধীনভাবে GrapheneOS সাইনিং কি (signing key) ডাউনলোড করবেন, ফ্যাক্টরি ইমেজ ফেচ করবেন এবং `ssh-keygen` ব্যবহার করে এর অথেন্টিসিটি ক্রিপ্টোগ্রাফিক্যালি ভেরিফাই করবেন — যা নিশ্চিত করে যে ট্রানজিটের সময় ডাউনলোড ইন্টারসেপ্ট করা হলেও, কোনো ট্যাম্পারড ইমেজ ম্যাথমেটিক্যালি রিজেক্ট হয়ে যাবে।

প্রথমে, পাবলিক সাইনিং কি ডাউনলোড করুন:

```
curl -O https://releases.grapheneos.org/allowed_signers
```

এই ফাইলের কন্টেন্ট হওয়া উচিত:

```
contact@grapheneos.org ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIIUg/m5CoP83b0rfSCzYSVA4cw4ir49io5GPoxbgxdJE
```

ইন্ডিপেনডেন্ট ভেরিফিকেশনের জন্য আপনি Bluesky, X, এবং GitHub-এ GrapheneOS-এর অফিসিয়াল পোস্টগুলোর সাথে এই কি-টির ক্রস-রেফারেন্স করতে পারেন।

এরপর, আপনার নির্দিষ্ট ডিভাইসের জন্য ফ্যাক্টরি ইমেজ এবং তার সিগনেচার ডাউনলোড করুন। `DEVICE_NAME`-এর জায়গায় আপনার ডিভাইসের কোডনেম (যেমন, Pixel 8 Pro-এর জন্য `husky`) এবং `VERSION`-এর জায়গায় GrapheneOS releases page থেকে লেটেস্ট রিলিজ ভার্সনটি বসিয়ে নিন:

```
curl -O https://releases.grapheneos.org/DEVICE_NAME-install-VERSION.zip
curl -O https://releases.grapheneos.org/DEVICE_NAME-install-VERSION.zip.sig
```

এবার সিগনেচার ভেরিফাই করুন। **Linux এবং macOS-এ:**

```
ssh-keygen -Y verify -f allowed_signers -I contact@grapheneos.org -n "factory
images" -s DEVICE_NAME-install-VERSION.zip.sig < DEVICE_NAME-install-
VERSION.zip
```

Windows (PowerShell)-এ:

```
cmd /c 'ssh-keygen -Y verify -f allowed_signers -I contact@grapheneos.org -n "factory images" -s DEVICE_NAME-install-VERSION.zip.sig < DEVICE_NAME-install-VERSION.zip'
```

সফল ভেরিফিকেশনের আউটপুট হবে:

```
Good "factory images" signature for contact@grapheneos.org with ED25519 key SHA256:AhgHif0mei+9aNyKLfMZBh2yptHdw/aN7Tlh/j2eFwM
```

আউটপুটে যদি `Good` ছাড়া অন্য কিছু দেখায়, তবে **অবিলম্বে থেমে যান**। ফাইলটি করাপ্টেড বা ট্যাম্পারড হয়ে থাকতে পারে। এটি পুনরায় ডাউনলোড করুন এবং আবার চেষ্টা করুন। ফেইলড ভেরিফিকেশন নিয়ে কখনোই সামনে অগ্রসর হবেন না।

ধাপ ৫: ফ্যাক্টরি ইমেজ এক্সট্র্যাক্ট এবং ফ্ল্যাশ করা

ইমেজ ভেরিফাই করা হয়ে গেলে, আর্কাইভটি এক্সট্র্যাক্ট করুন:

Linux-এ:

```
bsdtar xvf DEVICE_NAME-install-VERSION.zip
```

macOS এবং Windows-এ:

```
tar xvf DEVICE_NAME-install-VERSION.zip
```

এক্সট্র্যাক্ট করা ডিরেক্টরিতে নেভিগেট করুন এবং ফ্ল্যাশ স্ক্রিপ্ট এক্সিকিউট করুন:

Linux এবং macOS-এ:

```
cd DEVICE_NAME-install-VERSION
bash flash-all.sh
```

Windows-এ:

```
cd DEVICE_NAME-install-VERSION
./flash-all.bat
```

স্ক্রিপ্টটি স্বয়ংক্রিয়ভাবে মাল্টি-স্টেজ প্রসেস হ্যান্ডেল করবে: আপডেটেড ফার্মওয়্যার ফ্ল্যাশ করা, bootloader ইন্টারফেসে রিবুট করা এবং GrapheneOS অপারেটিং সিস্টেম পার্টিশন ফ্ল্যাশ করা। **এই সময়ে ডিভাইসটি স্পর্শ করবেন না বা ক্যাবল ডিসকানেক্ট করবেন না।** স্ক্রিপ্টটি সম্পন্ন হওয়া পর্যন্ত ধৈর্য সহকারে অপেক্ষা করুন।

> ***Troubleshooting (Linux):*** *যদি স্টোরেজ এররের কারণে স্ক্রিপ্টটি ফেইল করে, তবে আপনার `/tmp` ডিরেক্টরিটি হয়তো খুব ছোট। একটি লোকাল টেম্পোরারি ডিরেক্টরি তৈরি করুন এবং পুনরায় রান করুন:*
>
> ```
> mkdir tmp && TMPDIR="$PWD/tmp" bash flash-all.sh
> ```

ধাপ ৬: Bootloader লক করা

সম্পূর্ণ প্রসেসের মধ্যে এটি সবচেয়ে ক্রিটিক্যাল কমান্ড। ফ্ল্যাশিং স্ক্রিপ্ট কমপ্লিট হওয়ার পর এবং ডিভাইসটি Fastboot Mode-এ ফিরে এলে, bootloader লক করুন:

```
fastboot flashing lock
```

Volume বাটন দিয়ে "Lock the bootloader"-এ নেভিগেট করে এবং Power বাটন প্রেস করে ডিভাইসে কনফার্ম করুন। ডিভাইসটি একটি দ্বিতীয় ফ্যাক্টরি রিসেট পারফর্ম করবে।

bootloader লক করার মাধ্যমে **ফুল Verified Boot** অ্যাক্টিভেট হয়। এটি ডিভাইসের সিকিউর এলিমেন্টে GrapheneOS ক্রিপ্টোগ্রাফিক সাইনিং কি এনরোল করে। এরপর থেকে, প্রতিটি বুটে, হার্ডওয়্যার ম্যাথমেটিক্যালি সম্পূর্ণ অপারেটিং সিস্টেমের ইন্টিগ্রিটি ভেরিফাই করবে। এটি যেকোনো পার্টিশন ফ্ল্যাশ, ফরম্যাট বা ইরেজ করার জন্য `fastboot` ব্যবহারের সক্ষমতা স্থায়ীভাবে ডিজেবল করে দেয় — যা আনঅথোরাইজড ফার্মওয়্যার মডিফিকেশনের বিরুদ্ধে ডিভাইসটিকে সিল করে দেয়।

ধাপ ৭: পোস্ট-ইনস্টলেশন

GrapheneOS-এ বুট করার জন্য Power বাটন প্রেস করুন। ফাইনাল সেটআপ স্ক্রিনে, নিশ্চিত করুন যে **disable OEM unlocking** টগলটি চেক করা আছে (ডিফল্টভাবেই এটি থাকে)। এটি একজন অ্যাটাকারকে রানিং অপারেটিং সিস্টেমে প্রথমে অথেনটিকেট করা ছাড়াই bootloader-এ পুনরায় প্রবেশ করা থেকে প্রিভেন্ট করে।

আপনার ডিভাইসটি এখন একটি ক্রিপ্টোগ্রাফিক্যালি সিলড, ভেরিফাইড-বুট-এনাবেলড ফোর্ট্রেস — যা সম্পূর্ণভাবে আপনার ডিরেক্ট টার্মিনাল কন্ট্রোলের অধীনে ইনস্টল করা হয়েছে। প্রতিটি বাইট আপনার নিজের হাতে ডাউনলোড, ভেরিফাই এবং ফ্ল্যাশ করা হয়েছে। এখানে ব্রাউজারের ওপর ট্রাস্ট ডেলিগেশন নেই, কোনো অ্যাবস্ট্রাকশন লেয়ার নেই এবং কোনো অ্যাম্বিগুয়িটি নেই। এটাই একজন অপারেটরের সঠিক পাথ।

১.৬ পোস্ট-ইনস্টলেশন: Verified Boot এবং Auditor অ্যাপ

অপারেটিং সিস্টেম ফ্ল্যাশ করা হয়েছে, এবং bootloader ক্রিপ্টোগ্রাফিক্যালি সিল করা হয়েছে। আপনার ডিভাইসটি যখন প্রথমবারের মতো চালু হবে, তখন আপনি GrapheneOS-এর কোর সিকিউরিটি মেকানিজমগুলোকে প্রাণবন্ত হতে দেখতে পাবেন। এই পোস্ট-ইনস্টলেশন ফেজটি আপনার নতুন ফোর্ট্রেস বা দুর্গের ইন্টিগ্রিটি ভেরিফাই করার জন্য এবং ফ্যাক্টরি থেকে আপনার হাতে পৌঁছানো পর্যন্ত হার্ডওয়্যারটিতে কোনো ফিজিক্যাল বা ডিজিটাল ট্যাম্পারিং বা কারচুপি হয়নি তা নিশ্চিত করার জন্য ডেডিকেটেড।

হলুদ ওয়ার্নিং স্ক্রিন

পাওয়ার বাটন প্রেস করার সাথে সাথেই আপনাকে একটি উজ্জ্বল হলুদ ওয়ার্নিং স্ক্রিন দ্বারা স্বাগত জানানো হবে, যেখানে বলা থাকবে: *"Your device is loading a different operating system."*

একজন নতুন ইউজারের কাছে, এই মেসেজটি অনেক সময় তাৎক্ষণিক প্যানিক ট্রিগার করে। এটিকে একটি এরর মনে হলেও, আসলে এটি সম্পূর্ণ বিপরীত। এই হলুদ স্ক্রিনটি হলো আপনার ডিভাইসের ফিজিক্যাল সিলিকন যা সফলভাবে এর সিকিউরিটি প্রোটোকল এক্সিকিউট করছে। এটি হলো ভিজ্যুয়াল প্রমাণ যে **Android Verified Boot (AVB)** সক্রিয়ভাবে কাজ করছে।

> ***সিকিউরিটি ডেফিনিশন: Android Verified Boot (AVB)*** *Verified Boot একটি ইমিউটেবল হার্ডওয়্যার রুট অফ ট্রাস্ট (root of trust) থেকে ফার্মওয়্যার এবং সম্পূর্ণ অপারেটিং সিস্টেমের (উভয়ই কার্নেল এবং ইউজারস্পেস) ইন্টিগ্রিটি এবং অথেন্টিসিটি ভ্যালিডেট করে। প্রতিবার ডিভাইসটি বুট হওয়ার সময়, হার্ডওয়্যার অপারেটিং সিস্টেমটিকে রান করার অনুমতি দেওয়ার আগে ম্যাথমেটিক্যালি এর সিগনেচার ভেরিফাই করে, যা নিশ্চিত করে যে এটি কোনো ম্যালওয়্যার বা আনঅথোরাইজড মডিফিকেশন দ্বারা কম্প্রোমাইজড হয়নি।*

স্ট্যান্ডার্ড কমার্শিয়াল ডিভাইসগুলোতে হার্ডওয়্যারের ভেতরে Google বা ম্যানুফ্যাকচারারের ক্রিপ্টোগ্রাফিক কি (keys) হার্ডকোড করা থাকে। ডিভাইসটি যখন স্টক OS লোড করে, তখন এই কি-গুলো নীরবে ম্যাচ হয়ে যায়। যেহেতু আপনি GrapheneOS-এর কাস্টম ক্রিপ্টোগ্রাফিক কি দিয়ে bootloader লক করেছেন, সেহেতু হার্ডওয়্যার ফ্যাক্টরি-ডিফল্ট কি এবং নতুন যে কি-টিকে ট্রাস্ট করতে বলা হয়েছে তার মধ্যে একটি ডিসক্রিপেন্সি (discrepancy) বা অসঙ্গতি ডিটেক্ট করে। এই হলুদ স্ক্রিনটি মূলত হার্ডওয়্যার কর্তৃক আপনাকে বিশ্বস্ততার সাথে ইনফর্ম করা যে একটি বিকল্প, ইউজার-ইনস্টলড অপারেটিং সিস্টেম বুট হচ্ছে।

আধুনিক Pixel ডিভাইসগুলোতে (Pixel 6 এবং তার পরের), এই স্ক্রিনটি একটি ক্রিপ্টোগ্রাফিক হ্যাশ (hash) প্রদর্শন করবে। আপনি GrapheneOS ওয়েবসাইটে প্রকাশিত অফিসিয়াল হ্যাশগুলোর সাথে এই হ্যাশটিকে ভিজ্যুয়ালি ক্রস-রেফারেন্স করতে পারেন। যদি এগুলো নিখুঁতভাবে ম্যাচ করে, তবে আপনার ইনস্টলেশন ম্যাথমেটিক্যালি ফ্ললেস বা ত্রুটিমুক্ত।

ফার্স্ট সেটআপ এবং OEM Unlocking ডিজেবল করা

ওয়ার্নিং স্ক্রিনটি সরে যাওয়ার পর, আপনি প্রাথমিক GrapheneOS সেটআপ উইজার্ডে প্রবেশ করবেন। যেহেতু GrapheneOS সম্পূর্ণভাবে ডি-গুগলড (de-Googled), তাই এই প্রসেসটি অবিশ্বাস্যভাবে সংক্ষিপ্ত। এখানে সাইন ইন করার মতো কোনো Google অ্যাকাউন্ট নেই, ডিক্লাইন করার মতো কোনো টেলিমেট্রি অ্যাগ্রিমেন্ট নেই, এবং ক্লাউডের কাছে সারেন্ডার করার মতো কোনো বায়োমেট্রিক ডেটাও নেই।

আপনার টাইমজোন সেট করতে এবং একটি সিকিউর PIN বা পাসফ্রেজ কনফিগার করার জন্য স্ট্যান্ডার্ড প্রম্পটগুলোর মধ্য দিয়ে অগ্রসর হোন। সেটআপ উইজার্ডের ফাইনাল স্ক্রিনে, আপনি **OEM unlocking** সম্পর্কিত একটি অত্যন্ত ক্রিটিক্যাল সিকিউরিটি টগলের সম্মুখীন হবেন।

ডিফল্টরূপে, **disable OEM unlocking** করার জন্য এই বক্সটি চেক করা থাকে। আপনাকে অবশ্যই এটি চেক করা অবস্থায় রাখতে হবে।

প্রি-ইনস্টলেশন ফেজে, আপনি হার্ডওয়্যার ডিফেন্স লোয়ার করতে এবং OS ফ্ল্যাশ করতে OEM unlocking এনাবেল করেছিলেন। এখন যেহেতু আপনি বাঙ্কারের ভেতরে আছেন এবং দরজা লক করা আছে, আপনাকে অবশ্যই সেই মেকানিজমটি ডিজেবল করতে হবে যা প্রথমে আপনাকে দরজাটি খোলার অনুমতি দিয়েছিল। এখানে OEM unlocking ডিজেবল করার মাধ্যমে, আপনি এটি নিশ্চিত করেন যে যদি কোনো অ্যাডভার্সারি (adversary) আপনার ডিভাইসে ফিজিক্যাল অ্যাক্সেস পেয়েও যায়, তবুও তারা আপনার রানিং অপারেটিং সিস্টেমে আপনার PIN দিয়ে প্রথমে অথেনটিকেট করা ছাড়া bootloader-এ রিবুট করে ফোন ওয়াইপ করতে পারবে না বা ম্যালিশিয়াস ফার্মওয়্যার ফ্ল্যাশ করার চেষ্টাও করতে পারবে না।

Auditor অ্যাপ: হার্ডওয়্যার-ব্যাকড অ্যাটেস্টেশন (Hardware-Backed Attestation)

ইনস্টলেশন সম্পূর্ণ হয়েছে, কিন্তু এলিট সাইবারসিকিউরিটির জগতে, আমরা ট্রাস্ট বা বিশ্বাসের ওপর নির্ভর করি না; আমরা ক্রিপ্টোগ্রাফিক প্রমাণের ওপর নির্ভর করি।

আপনি কীভাবে পুরোপুরি নিশ্চিত হতে পারেন যে GrapheneOS ইনস্টল করার জন্য আপনি যে কম্পিউটারটি ব্যবহার করেছেন সেটি কম্প্রোমাইজড ছিল না? আপনি কীভাবে জানেন যে কোনো অ্যাডভান্সড স্টেট-লেভেল অ্যাক্টর শিপিংয়ের সময় আপনার ফোনটি ইন্টারসেপ্ট করেনি এবং এমন একটি হাইলি সফিস্টিকেটেড সাপ্লাই-চেইন অ্যাটাক ইমপ্ল্যান্ট করেনি যা হুবহু GrapheneOS-কে মিমিক বা অনুকরণ করে?

এই থ্রেটগুলোর (threats) জবাব দেওয়ার জন্য, GrapheneOS ফাউন্ডেশন **Auditor** অ্যাপটি তৈরি করেছে।

> ***সিকিউরিটি ডেফিনিশন: হার্ডওয়্যার-ব্যাকড অ্যাটেস্টেশন (Hardware-Backed Attestation)*** *অ্যাটেস্টেশন হলো ডিভাইসের হার্ডওয়্যারের ফিজিক্যালি আইসোলেটেড, সিকিউর এলিমেন্টগুলো (যেমন Titan M2 চিপ বা StrongBox Keymaster) ব্যবহার করে একটি ক্রিপ্টোগ্রাফিক সিগনেচার তৈরি করার প্রসেস যা সফটওয়্যারের বর্তমান স্টেট বা অবস্থাকে ভেরিফাই করে। এই সিগনেচার একজন এক্সটার্নাল অবজার্ভারের কাছে প্রমাণ করে যে OS-টি জেনুইন, bootloader লক করা আছে এবং কোনো ট্যাম্পারিং বা কারচুপি হয়নি।*

Auditor অ্যাপটি Trust On First Use (TOFU) মডেল লিভারেজ (leverage) করে। এটি যাচাই করা হচ্ছে এমন ডিভাইসটি ("Auditee") এবং যাচাই করার কাজে ব্যবহৃত দ্বিতীয় একটি পৃথক Android ডিভাইসের ("Auditor") মধ্যে একটি পেয়ারিং (pairing) প্রসেস সম্পন্ন করে।

কীভাবে লোকাল ভেরিফিকেশন পারফর্ম করবেন

এই চেকটি পারফর্ম করার জন্য, আপনার একটি কার্যকর ক্যামেরাযুক্ত সেকেন্ডারি Android ডিভাইস (যেটিতে Android 13 বা তার ওপরের ভার্সন রান করছে) প্রয়োজন হবে। এটিতে GrapheneOS রান করার কোনো প্রয়োজন নেই; একটি স্ট্যান্ডার্ড Android ফোন বা ট্যাবলেট এতে পুরোপুরি কাজ করবে।

ধাপ ১: Auditor অ্যাপ ইনস্টল করা আপনার নতুন ইনস্টল করা GrapheneOS ডিভাইসে (Auditee), প্রি-ইনস্টলড "Apps" রিপোজিটরিটি খুলুন এবং **Auditor** অ্যাপটি ইনস্টল করুন। আপনার সেকেন্ডারি Android ডিভাইসে (Auditor), Google Play Store থেকে অ্যাপটি ডাউনলোড করুন (`app.attestation.auditor.play` লিখে সার্চ করুন)।

ধাপ ২: রোল (Roles) ইনিশিয়েট করা উভয় ডিভাইসেই Auditor অ্যাপটি ওপেন করুন।

- আপনার সেকেন্ডারি ডিভাইসে, "**Auditor**" লেবেলযুক্ত বাটনটিতে ট্যাপ করুন। এটি একটি QR কোড প্রদর্শন করবে।
- আপনার নতুন GrapheneOS ডিভাইসে, "**Auditee**" লেবেলযুক্ত বাটনটিতে ট্যাপ করুন। এটি ক্যামেরা ভিউফাইন্ডারটি ওপেন করবে।

ধাপ ৩: ফার্স্ট হ্যান্ডশেক (চ্যালেঞ্জ) আপনার GrapheneOS ডিভাইসের ক্যামেরাটি সেকেন্ডারি ডিভাইসে প্রদর্শিত QR কোডটির দিকে পয়েন্ট করুন। এটি Auditor থেকে Auditee-তে একটি ক্রিপ্টোগ্রাফিক চ্যালেঞ্জ ট্রান্সফার করে। আপনার GrapheneOS ডিভাইসের ভেতরের সিকিউর এলিমেন্ট এখন এর সফটওয়্যার ইন্টিগ্রিটি, bootloader স্ট্যাটাস এবং ডাউনগ্রেড প্রোটেকশন স্টেটের একটি ম্যাথমেটিক্যাল প্রুফ তৈরি করবে।

ধাপ ৪: Auditor-কে অ্যাডভান্স করা আপনার GrapheneOS ডিভাইসটি QR কোড রিড করার পর, এর স্ক্রিন পরিবর্তিত হয়ে নিজস্ব একটি QR কোড প্রদর্শন করবে। আপনার সেকেন্ডারি ডিভাইসে (Auditor), অ্যাপটিকে অ্যাডভান্স করতে স্ক্রিনে থাকা QR কোডটিতে ট্যাপ করুন। এটি এখন তার নিজস্ব ক্যামেরা ভিউফাইন্ডার ওপেন করবে।

ধাপ ৫: সেকেন্ড হ্যান্ডশেক (ভেরিফিকেশন) সেকেন্ডারি ডিভাইসের ক্যামেরাটি এখন আপনার GrapheneOS ডিভাইসে প্রদর্শিত QR কোডটির দিকে পয়েন্ট করুন। সেকেন্ডারি ডিভাইসটি আপনার ফোনের হার্ডওয়্যার দ্বারা জেনারেট করা ক্রিপ্টোগ্রাফিক প্রুফটি রিড করবে।

ধাপ ৬: অ্যাটেস্টেশন রেজাল্ট রিভিউ করা সেকেন্ডারি ডিভাইসটি Auditee-এর সিকিউর এলিমেন্ট (Titan M2 / StrongBox Keymaster) দ্বারা উৎপাদিত হার্ডওয়্যার-জেনারেটেড সার্টিফিকেট চেইন ভেরিফাই করবে। প্রথম পেয়ারিংয়ের সময়, Auditor Trust On First Use মডেল ব্যবহার করে ডিভাইসটির হার্ডওয়্যার-ব্যাকড আইডেন্টিটিকে পিন (pin) করে। পরবর্তী সকল ভেরিফিকেশনের সময়, এটি কনফার্ম করে যে একই ফিজিক্যাল হার্ডওয়্যার একই ক্রিপ্টোগ্রাফিক আইডেন্টিটি প্রেজেন্ট করছে কি না। স্ক্রিনটি সবুজ হয়ে যাবে এবং একটি সফল ভেরিফিকেশন রেজাল্ট প্রদর্শন করবে। এটি কনফার্ম করবে:

- ডিভাইস আইডেন্টিটি এবং হার্ডওয়্যার মডেল।
- অপারেটিং সিস্টেমটি জেনুইন GrapheneOS বিল্ড।
- Android Verified Boot অ্যাক্টিভ আছে এবং bootloader স্থায়ীভাবে লক করা আছে।
- অপারেটিং সিস্টেমটিকে গোপনে কোনো ভালনারেবল (vulnerable) পুরোনো ভার্সনে ডাউনগ্রেড করা হয়নি।

হাই-রিস্ক অপারেটরদের জন্য শিডিউলড রিমোট ভেরিফিকেশন

লোকাল পেয়ারিং প্রসেসটি তাৎক্ষণিক মানসিক প্রশান্তি দিলেও, হাই-রিস্ক অপারেটররা—যেমন ইনভেস্টিগেটিভ জার্নালিস্ট, ডিসিডেন্ট (dissidents) বা এন্টারপ্রাইজ ফ্লিট ম্যানেজারদের—প্রায়শই একটি চলমান নিশ্চয়তার প্রয়োজন হয় যে কোনো আন্তর্জাতিক সীমান্ত অতিক্রমের সময় বা ফিজিক্যাল সিজারের (seizure) সময় তাদের ডিভাইসটি গোপনে কম্প্রোমাইজড হয়নি।

এর জন্য, Auditor অ্যাপটি `attestation.app` সার্ভিসের মাধ্যমে অটোমেটেড, রিমোট ভেরিফিকেশন সাপোর্ট করে।

একটি পৃথক, সিকিউর কম্পিউটার থেকে Attestation প্ল্যাটফর্মে অ্যাকাউন্ট তৈরি করার মাধ্যমে, আপনি আপনার GrapheneOS ডিভাইসকে নিয়মিত শিডিউল অনুযায়ী নীরবে হার্ডওয়্যার-ব্যাকড অ্যাটেস্টেশন চেক পারফর্ম করার এবং সার্ভারে ক্রিপ্টোগ্রাফিক প্রুফগুলো সাবমিট করার নির্দেশ দিতে পারেন। যদি আপনার ডিভাইসটি কোনো চেক-ইন মিস করে, বা সার্ভার ডিটেক্ট করে যে ভেরিফাইড বুট স্টেটটি অল্টারড বা ডাউনগ্রেডেড হয়েছে, তবে সার্ভিসটি অবিলম্বে আপনার নির্ধারিত ঠিকানায় একটি ইমার্জেন্সি ইমেইল অ্যালার্ট পাঠিয়ে দেবে। এটি মূলত ডিভাইস ইন্টিগ্রিটির জন্য একটি ডিজিটাল "ডেড ম্যান'স সুইচ" (dead man's switch) হিসেবে কাজ করে।

এই ভেরিফিকেশন প্রসেসটি সম্পন্ন করার মাধ্যমে, আপনি একটি মিলিটারি-গ্রেড সাপ্লাই-চেইন অডিট এক্সিকিউট করেছেন। আপনি ম্যাথমেটিক্যালি প্রমাণ করেছেন যে আপনার হাতের সিলিকনটি সেই প্রাইভেসি ফোর্ট্রেসের একটি আনট্যাম্পারড (untampered) বা অটুট, জেনুইন ভার্সন রান করছে যা আপনি তৈরি করতে চেয়েছিলেন।

এখন আপনি Auditor অ্যাপটি বন্ধ করতে পারেন এবং নিশ্চিন্তে থাকতে পারেন। ফাউন্ডেশন তৈরি হয়ে গেছে, দেয়ালগুলো ভেরিফাই করা হয়েছে এবং বাঙ্কারটি অপারেশনাল।

১.৭ ফার্স্ট বুট: UI এবং Settings নেভিগেট করা

আপনি যখন প্রথমবারের মতো ডিভাইসটি আনলক করবেন, তখন আপনি যা দেখবেন তা দেখে হয়তো অবাক হতে পারেন। বা বলা ভালো, আপনি যা দেখবেন না তা দেখে।

স্ট্যান্ডার্ড কমার্শিয়াল স্মার্টফোনগুলো আউট-অফ-দ্য-বক্স (out of the box) অ্যাগ্রেসিভলি ক্লাটারড (cluttered) বা বিশৃঙ্খল থাকে। সেগুলো ডিলিট-অযোগ্য ক্যারিয়ার অ্যাপ্লিকেশন, প্রোপ্রাইটারি ম্যানুফ্যাকচারার "স্কিন" (skins), ওভারল্যাপিং অ্যাপ স্টোর এবং ব্যাকগ্রাউন্ড সার্ভিস দিয়ে বোঝাই থাকে, যা নীরবে দূরবর্তী সার্ভারগুলোতে পিং (ping) করতে থাকে।

GrapheneOS এর সম্পূর্ণ ও ইচ্ছাকৃত বিপরীত। আপনি যে ইন্টারফেসটি দেখছেন তা হলো পিওর, আনঅ্যাডাল্টারেটেড Android Open Source Project (AOSP) কোড। এটি অ্যাগ্রেসিভলি মিনিমালিস্ট এবং ব্লোটওয়্যার (bloatware) থেকে সম্পূর্ণ মুক্ত। আপনি এখানে কেবল একটি ফোন ডায়ালার, একটি মেসেজিং অ্যাপ, একটি ক্যামেরা, একটি ক্যালকুলেটর, একটি সেটিংস মেনু এবং Vanadium ওয়েব ব্রাউজার পাবেন।

এখানে আপনার হোম স্ক্রিনে স্থায়ীভাবে অ্যাঙ্কর করা কোনো Google Search বার নেই। আপনার ইন্টারেস্ট ট্র্যাক করার মতো কোনো নিউজ ফিড নেই। এই অপারেটিং সিস্টেমটি একটি শূন্য, ম্যাথমেটিক্যালি সিকিউর ক্যানভাস, যা আপনার প্রিসাইস অপারেশনাল কমান্ডের অপেক্ষায় রয়েছে।

অ্যাপ্লিকেশন ইনস্টল বা নেটওয়ার্ক ডিফেন্স কনফিগার শুরু করার আগে, আমাদেরকে অবশ্যই বেইসলাইন ইন্টারেকশন প্রোটোকলগুলো এস্টাবলিশ করতে হবে এবং কয়েকটি ক্রিটিক্যাল হার্ডওয়্যার সেটিংস লক ডাউন করতে হবে।

Gesture Navigation বনাম 3-Button Navigation

অপারেশনাল ফ্লুইডিটির (fluidity) জন্য একটি আধুনিক মোবাইল OS এফিশিয়েন্টলি নেভিগেট করা অত্যন্ত ক্রিটিক্যাল। ডিফল্টরূপে, GrapheneOS-এ Gesture Navigation ব্যবহার করা হয়। আপনি যদি কোনো পুরোনো Android ডিভাইস থেকে মাইগ্রেট করে থাকেন, তবে আপনি হয়তো ট্র্যাডিশনাল 3-Button Navigation-এ (স্ক্রিনের নিচে অ্যাঙ্কর করা Back, Home এবং Recent Apps বাটন) অভ্যস্ত।

যদিও আপনি সেটিংস থেকে (**Settings** > **System** > **Navigation mode**) পুনরায় 3-Button Navigation-এ ফিরে যেতে পারেন, তবে আমরা অপারেটরদের Gesture Navigation আয়ত্ত করার জন্য দৃঢ়ভাবে উৎসাহিত করি।

জেসচারগুলো স্ট্যাটিক নেভিগেশন বারটি সরিয়ে দেয়, যার ফলে আপনি স্ক্রিনের 100% রিয়েল এস্টেট ব্যবহার করতে পারেন। আরও গুরুত্বপূর্ণ বিষয় হলো, জেসচারগুলো উল্লেখযোগ্যভাবে ফাস্ট এবং যেকোনো জটিল পরিস্থিতিতে ফ্লুইড, ওয়ান-হ্যান্ডেড অপারেশন (one-handed operation) অ্যালাউ করে।

- **Home-এ যাওয়ার জন্য:** স্ক্রিনের একেবারে নিচের প্রান্ত থেকে দ্রুত ওপরের দিকে সোয়াইপ করুন।
- **Recent Apps দেখার জন্য:** নিচের প্রান্ত থেকে ওপরের দিকে সোয়াইপ করুন এবং স্ক্রিনে আপনার আঙুলটি এক সেকেন্ডের ভগ্নাংশের জন্য ধরে রাখুন।
- **Back-এ যাওয়ার জন্য:** স্ক্রিনের ডান বা বাম প্রান্ত থেকে ভেতরের দিকে সোয়াইপ করুন।

মাসল মেমোরি তৈরি হতে হয়তো এক বা দুই দিন সময় লাগতে পারে, কিন্তু একবার অভ্যস্ত হয়ে গেলে এর এফিশিয়েন্সি অতুলনীয়।

Background রেডিও ট্রান্সমিশন সিকিউর করা

আপনার ডিভাইসে একাধিক রেডিও (Wi-Fi, Bluetooth, Cellular, NFC) রয়েছে। এমনকি আপনি যদি অ্যাক্টিভলি কোনো Wi-Fi নেটওয়ার্ক বা Bluetooth হেডসেট ব্যবহার নাও করেন, স্ট্যান্ডার্ড কমার্শিয়াল অপারেটিং সিস্টেমগুলো প্রতিনিয়ত ব্যাকগ্রাউন্ডে কাছাকাছি অ্যাক্সেস পয়েন্টগুলোর (access points) জন্য স্ক্যান করতে থাকে।

টেক কনগ্লোমারেটগুলো (tech conglomerates) এই ব্যাকগ্রাউন্ড স্ক্যানিং ব্যবহার করে আপনার ফিজিক্যাল লোকেশনকে ভয়ংকর অ্যাকুরেসি বা নির্ভুলতার সাথে ম্যাপ করে। আপনি যদি কোনো কফি শপের পাশ দিয়ে হেঁটে যান, আপনার ফোন তাদের Wi-Fi রাউটার দেখতে পায়। OS সেই রাউটারের ইউনিক MAC অ্যাড্রেস লগ করে, সেটিকে কফি শপটির GPS কোঅর্ডিনেটগুলোর (coordinates) সাথে কোরিলেট (correlate) করে, এবং তাদের সার্ভারে আপনার লোকেশন প্রোফাইল আপডেট করে—এমনকি টেকনিক্যালি আপনার GPS "off" থাকা অবস্থাতেও।

GrapheneOS এই ডেটা ক্লাউডে রিপোর্ট করে না, কিন্তু কাছাকাছি ফিজিক্যাল অবজার্ভারদের কাছে আপনার উপস্থিতি ব্রডকাস্ট করাটা একটি পুওর Operational Security (OpSec)। আমাদেরকে অবশ্যই নিশ্চিত করতে হবে যে হার্ডওয়্যারটি যখন এক্সপ্লিসিটলি (explicitly) ব্যবহার করা হচ্ছে না তখন তা সম্পূর্ণ সাইলেন্ট থাকে।

১. **Settings > Location > Location services**-এ নেভিগেট করুন। ২. **Wi-Fi scanning** ট্যাপ করুন এবং এটি **OFF** করে দিন। ৩. **Bluetooth scanning** ট্যাপ করুন এবং এটি **OFF** করে দিন।

যখন আপনার Wi-Fi বা Bluetooth প্রয়োজন হবে, তখন আপনি ম্যানুয়ালি সেগুলো অন করবেন। কাজ শেষ হলে অফ করে দেবেন। আপনার এক্সপ্লিসিট অর্ডার ছাড়া ডিভাইসের কখনোই কোনো বীকন (beacon) ট্রান্সমিট করা উচিত নয়।

অথেনটিকেটশন হার্ডেনিং: পাসওয়ার্ড এবং PIN Scrambling

ইনিশিয়াল সেটআপ উইজার্ডের সময়, আপনাকে একটি PIN বা পাসওয়ার্ড তৈরি করার জন্য প্রম্পট করা হয়েছিল। আপনি যদি সুবিধার জন্য একটি 4-ডিজিটের PIN সেট করে থাকেন, তবে আপনাকে অবশ্যই তা এখনই পরিবর্তন করতে হবে।

আগের সেকশনগুলোতে যেমনটা আলোচনা করা হয়েছে, আপনার GrapheneOS ডিভাইসটি একটি ডেডিকেটেড সিকিউর এলিমেন্ট (Titan M2 বা StrongBox) দ্বারা প্রটেক্টেড। এই চিপটি Weaver API সাপোর্ট করে, যা ব্রুট-ফোর্স (brute-force) পাসকোড গেসিংকে ফিজিক্যালি থ্রটল (throttle) করে। যদি কোনো অ্যাডভার্সারি (adversary) আপনার ফোনকে কোনো ফরেনসিক এক্সট্র্যাকশন মেশিনে (forensic extraction machine) প্লাগ ইন করে, তবে সিকিউর এলিমেন্ট ইচ্ছাকৃতভাবে হার্ডওয়্যারটিকে স্লো করে দেবে, যার ফলে মেশিনটি প্রতিটি গেস বা অনুমানের মাঝখানে অপেক্ষা করতে বাধ্য হবে।

তবে, ম্যাথমেটিক্যাল রিয়্যালিটি এখনও এখানে অ্যাপ্লাই হয়। একটি 4-ডিজিটের PIN-এ মাত্র 10,000টি পসিবল কম্বিনেশন থাকে। হেভি হার্ডওয়্যার থ্রটলিং থাকা সত্ত্বেও, একটি 4-ডিজিটের PIN শেষ পর্যন্ত ব্রেক হয়ে যেতে পারে।

১. **Settings** > **Security** > **Screen lock**-এ নেভিগেট করুন। ২. আপনার লক মেথড হিসেবে **Password** সিলেক্ট করুন। ৩. লেটার (letters), নাম্বার এবং সিম্বল ব্যবহার করে একটি স্ট্রং, আলফানিউমেরিক (alphanumeric) পাসফ্রেজ তৈরি করুন। একটি র‍্যান্ডমলি জেনারেটেড 8 থেকে 12-ক্যারেক্টারের পাসফ্রেজ কার্যত ইমপেনেট্রেবল (impenetrable), এমনকি স্টেট-স্পন্সরড (state-sponsored) ফরেনসিক হার্ডওয়্যারের বিরুদ্ধেও।

PIN Scrambling এনাবেল করা

যদি আপনার থ্রেট মডেল (threat model) একটি PIN ব্যবহারের অনুমতি দেয় (হয়তো আপনি একটি অত্যন্ত দীর্ঘ, 8 থেকে 10-ডিজিটের PIN ব্যবহার করছেন), তবে আপনাকে অবশ্যই **PIN Scrambling** এনাবেল করতে হবে।

যখন আপনি মাসে শত শত বার একটি ফোনে স্ট্যাটিক PIN টাইপ করেন, তখন দুটি মারাত্মক ভালনারেবিলিটি বা দুর্বলতা দেখা দেয়: ১. **Shoulder Surfing:** ক্যাফে বা ট্রেনে আপনার আশেপাশে বসে থাকা কোনো অ্যাডভার্সারি খুব সহজেই নাম্বারগুলো টাইপ করার সময় আপনার বুড়ো আঙুলের জিওমেট্রিক প্যাটার্ন (geometric pattern) মেমোরাইজ করে ফেলতে

পারে। ২. **Smudge Analysis:** যদি আপনার ফোনটি সিজড (seized) বা চুরি হয়ে যায়, তবে একজন অ্যাডভার্সারি স্ক্রিনটিকে আলোর বিপরীতে ধরে গ্লাসের ওপরে থাকা ফিজিক্যাল ফিঙ্গারপ্রিন্ট স্মাজ (smudges) বা আঙুলের ছাপ পরীক্ষা করতে পারে। এই স্মাজগুলো স্পষ্টভাবে নির্দেশ করবে যে আপনি কোন নাম্বারগুলো সবচেয়ে বেশি প্রেস করেছেন।

GrapheneOS এর জন্য একটি ব্রিলিয়ান্ট এবং এলিগেন্ট মিটিগেশন (mitigation) প্রদান করে।

১. **Settings** > **Security** > **PIN settings**-এ নেভিগেট করুন (অথবা **Screen lock**-এর পাশের গিয়ার আইকনে ট্যাপ করুন)। ২. **Scramble PIN layout** টগলটি **ON** করে দিন।

এই ফিচারটি এনাবেল থাকলে, প্রতিবার স্ক্রিন অন হওয়ার সময় অপারেটিং সিস্টেম কিপ্যাডে নাম্বারগুলোর (0-9) পজিশন র‍্যান্ডমলি শাফল (shuffle) করবে। '1' নাম্বারটি একবার হয়তো নিচের ডান কোণায় থাকতে পারে, এবং পরের বার হয়তো ওপরের বাম কোণায় চলে যেতে পারে।

এটি স্মাজ অ্যানালাইসিসকে (smudge analysis) সম্পূর্ণভাবে নিউট্রালাইজ (neutralizes) করে দেয়, কারণ আপনি পুরো স্ক্রিন জুড়েই ট্যাপ করতে থাকবেন। এটি শোল্ডার সার্ফিংকেও (shoulder surfing) ব্যাপকভাবে মিটিগেট বা প্রশমিত করে, কারণ প্রতিটি আনলকের সময় আপনার বুড়ো আঙুলের জিওমেট্রিক প্যাটার্ন পরিবর্তন হয়ে যায়।

Biometrics এবং Fingerprint Unlock: এটি কি নিরাপদ?

GrapheneOS-এ Fingerprint (Biometrics) ব্যবহার করা নিরাপদ কি না, তা সম্পূর্ণ নির্ভর করে আপনার **Threat Model** বা আপনি কাদের হাত থেকে নিজেকে সুরক্ষিত রাখতে চাইছেন, তার ওপর।

টেকনিক্যাল দিক থেকে বিচার করলে GrapheneOS-এ ফিঙ্গারপ্রিন্ট ব্যবহার করা অত্যন্ত নিরাপদ। নিচে এর বিস্তারিত এবং অপসেক (OpSec) রিকমেন্ডেশন দেওয়া হলো:

টেকনিক্যাল সিকিউরিটি (The Good)

১. **ডেটা লোকাল থাকে:** আপনার ফিঙ্গারপ্রিন্ট ডেটা কখনোই ক্লাউডে (Google বা অন্য কোথাও) যায় না। এটি Pixel ডিভাইসের হার্ডওয়্যার-লেভেল সিকিউর এনক্লেভ বা Titan M2 চিপের ভেতরে এনক্রিপ্টেড অবস্থায় স্টোর করা থাকে। ২. **BFU (Before First Unlock):** ফোন রিস্টার্ট করার পর ডিভাইস BFU স্টেটে থাকে। তখন ফিঙ্গারপ্রিন্ট দিয়ে ফোন আনলক করা **অসম্ভব**। ফোন আনলক এবং ফাইল সিস্টেম ডিক্রিপ্ট করার জন্য আপনার অরিজিনাল PIN বা Password

দিতেই হবে। ৩. **Auto-Reboot ফিচার:** GrapheneOS-এ একটি Auto-reboot ফিচার আছে। আপনি যদি একটি নির্দিষ্ট সময় (যেমন ১২ বা ১৮ ঘণ্টা) ফোন আনলক না করেন, ফোনটি নিজে থেকেই রিস্টার্ট নিয়ে BFU স্টেটে চলে যায়। এরপর আর ফিঙ্গারপ্রিন্ট কাজ করবে না।

ফিজিক্যাল ও লিগ্যাল রিস্ক (The Bad)

ফিঙ্গারপ্রিন্টের সবচেয়ে বড় দুর্বলতা সফটওয়্যারে নয়, বরং ফিজিক্যাল রিয়েলিটিতে: **১. ফিজিক্যাল জবরদস্তি:** কেউ যদি আপনার ফোন কেড়ে নেয়, সে জোর করে আপনার আঙুল স্ক্যানারের ওপর রেখে ফোন আনলক করে ফেলতে পারে। ২. **লিগ্যাল ইস্যু:** বিশ্বের অনেক দেশে (যেমন ইউএসএ-এর ফিফথ অ্যামেন্ডমেন্ট অনুযায়ী) পুলিশ বা আইন-শৃঙ্খলা রক্ষাকারী বাহিনী আপনাকে আপনার পাসওয়ার্ড বা পিন বলতে বাধ্য করতে পারে না, কারণ সেটি আপনার মনের ভেতরে থাকে। কিন্তু তারা আইনিভাবে আপনাকে বাধ্য করতে পারে আঙুল দিয়ে বা ফেস আনলক দিয়ে ফোন খুলে দিতে।

রেকমেন্ডেশন: আপনার কী করা উচিত?

- **সাধারণ ব্যবহারকারীদের জন্য (Low-to-Medium Threat Model):** আপনি যদি হ্যাকার, ম্যালওয়্যার বা সাধারণ ফোন চুরি থেকে বাঁচতে চান, তবে **ফিঙ্গারপ্রিন্ট ব্যবহার করা সম্পূর্ণ নিরাপদ এবং সুবিধাজনক।** তবে শর্ত হলো—আপনার প্রাইমারি PIN বা Password-টি যেন খুবই শক্তিশালী (Alphanumeric) হয়।
- **ক্রিটিক্যাল থ্রেট মডেল (High Threat Model):** আপনি যদি এমন কোনো প্রফেশনে থাকেন যেখানে পুলিশি তল্লাশি, সরকারি নজরদারি বা কর্পোরেট গুপ্তচরবৃত্তির ঝুঁকি আছে (যেমন- ইনভেস্টিগেটিভ জার্নালিস্ট, পলিটিক্যাল অ্যাক্টিভিস্ট), তবে **ফিঙ্গারপ্রিন্ট সম্পূর্ণ বন্ধ রাখুন।** শুধুমাত্র একটি শক্তিশালী Alphanumeric Password ব্যবহার করুন।

> 💡 ***Pro Tip: Lockdown Mode:*** *আপনি যদি ফিঙ্গারপ্রিন্ট ব্যবহারও করেন, তবে GrapheneOS-এর* ***Lockdown*** *ফিচারটির কথা মাথায় রাখুন। আপনি যদি কখনো মনে করেন যে আপনার ফোন সিজ (seize) হতে পারে বা কেউ জোর করে আনলক করার চেষ্টা করবে, সাথে সাথে পাওয়ার বাটন চেপে ধরে "****Lockdown****"-এ ট্যাপ করুন। এটি মুহূর্তের মধ্যে আপনার ফিঙ্গারপ্রিন্ট ডিজেবল করে দেবে এবং ফোন আনলক করার জন্য পিন বা পাসওয়ার্ড দেওয়া বাধ্যতামূলক করে দেবে। সংক্ষেপে: দৈনন্দিন ব্যবহারের জন্য ফিঙ্গারপ্রিন্ট দারুণ একটি ব্যালেন্স (Security vs Convenience)। তবে ইমার্জেন্সিতে "Lockdown" অপশনটি ব্যবহার করার মানসিক প্রস্তুতি রাখুন।*

আপনার রেডিওগুলো সাইলেন্সড (silenced) এবং আপনার অথেনটিকেশন হার্ডেন (hardened) করার মাধ্যমে, আপনার ইন্টারফেসের ফাউন্ডেশনাল ডিফেন্স সম্পূর্ণ হলো। বাঙ্কারটি এখন ভেতর থেকে লক করা।

১.৮ বিল্ট-ইন (Built-in) GrapheneOS অ্যাপস

আপনি যখন একটি স্ট্যান্ডার্ড কমার্শিয়াল স্মার্টফোন কেনেন, তখন এর অপারেটিং সিস্টেমটি মূলত একটি স্টোরফ্রন্ট (storefront) হিসেবে কাজ করে। আপনাকে তাৎক্ষণিকভাবে ডুপ্লিকেট ওয়েব ব্রাউজার, ম্যানুফ্যাকচারার-ব্র্যান্ডেড মেসেজিং অ্যাপ, ডিলিট-অযোগ্য ক্যারিয়ার সফটওয়্যার এবং আপনার হোম স্ক্রিনে জায়গা পাওয়ার জন্য মোটা অঙ্কের টাকা প্রদান করা থার্ড-পার্টি অ্যাপ্লিকেশন দিয়ে বোম্বার্ড (bombarded) বা জর্জরিত করা হয়। এই প্রি-ইনস্টলড সফটওয়্যারগুলোকে সাধারণত "ব্লোটওয়্যার" (bloatware) বলা হয়। সাইবারসিকিউরিটির দৃষ্টিকোণ থেকে, ব্লোটওয়্যার শুধু একটি বিরক্তিই নয়—এটি আপনার অ্যাটাক সারফেসের (attack surface) একটি বিশাল সম্প্রসারণ। আপনার ডিভাইসে রান হওয়া কোডের প্রতিটি লাইনই এক্সপ্লয়টেশনের (exploitation) জন্য একটি সম্ভাব্য ভেক্টর (vector)।

GrapheneOS সম্পূর্ণ ভিন্ন একটি ফিলোসফি (philosophy) বা দর্শন গ্রহণ করে: **অ্যাবসোলিউট মিনিমালিজম (absolute minimalism)** বা চূড়ান্ত ন্যূনতমতা।

ডিভাইসটিকে একটি আধুনিক কমিউনিকেশন টুল হিসেবে কাজ করার জন্য অপারেটিং সিস্টেমটিতে শুধুমাত্র প্রয়োজনীয় এসেনশিয়াল (essential) অ্যাপ্লিকেশনগুলো অন্তর্ভুক্ত রয়েছে। আপনার ডিভাইসে থার্ড-পার্টি সফটওয়্যার হার্ড-ওয়্যার (hard-wire) করার জন্য GrapheneOS কোনো লোভনীয় পার্টনারশিপ করে না। এর পরিবর্তে, ডেভেলপাররা বেসপোক (bespoke), হাইলি হার্ডেনড (highly hardened) ডিফল্ট অ্যাপ্লিকেশনের একটি স্যুট (suite) তৈরি করেছেন। এগুলো কেবল AOSP (Android Open Source Project) ক্লোন নয়; এগুলো হলো ম্যাথমেটিক্যালি ফর্টিফায়েড (fortified) অ্যাপ্লিকেশন, যা যেকোনো হোস্টাইল ডিজিটাল এনভায়রনমেন্টে (hostile digital environments) অপারেট করার জন্য তৈরি করা হয়েছে।

চলুন আপনার জন্য উপলব্ধ কোর আর্সেনাল (core arsenal) বা মূল অস্ত্রভাণ্ডারটি পরীক্ষা করে দেখি।

Vanadium: হার্ডেনড ব্রাউজার

যেকোনো আধুনিক ডিভাইসে ওয়েব ব্রাউজারই সম্ভবত সবচেয়ে ক্রিটিক্যাল অ্যাপ্লিকেশন। এটি বাইরের দুনিয়ার সাথে আপনার প্রাইমারি ইন্টারফেস, এবং ফলস্বরূপ, এটি হলো রিমোট কোড এক্সিকিউশন (RCE) অ্যাটাকের প্রাইমারি ভেক্টর। প্রতিবার আপনি কোনো লিঙ্কে ট্যাপ করার সময়, আধুনিক ব্রাউজারগুলো কমপ্লেক্স JavaScript প্রসেস করে, ইন্ট্রিকেট (intricate) গ্রাফিক্স রেন্ডার করে এবং আনট্রাস্টেড (untrusted) সার্ভার থেকে ডায়নামিক কোড এক্সিকিউট করে।

GrapheneOS ওপেন-সোর্স Chromium রেন্ডারিং ইঞ্জিনের একটি হেভিলি হার্ডেনড ভ্যারিয়েন্ট (variant), **Vanadium**-এর মাধ্যমে এই থ্রেট মডেলটিকে অ্যাড্রেস করে। Vanadium কেবল একটি ব্রাউজার নয়; এটি পুরো অপারেটিং সিস্টেমের কোর `WebView` কম্পোনেন্ট, অর্থাৎ ওয়েব কনটেন্ট ডিসপ্লে করার প্রয়োজন হয় এমন যেকোনো অ্যাপ Vanadium-এর ফর্টিফায়েড স্যান্ডবক্সের (sandbox) মাধ্যমে তা রাউট (route) করবে।

Vanadium-কে কয়েকটি অ্যাডভান্সড মেকানিজমের মাধ্যমে জিরো-ডে এক্সপ্লয়টগুলো (zero-day exploits) মিটিগেট বা প্রশমিত করার জন্য তৈরি করা হয়েছে:

- **Strict Site Isolation:** Vanadium ডিপেস্ট বা গভীরতম লেভেলে প্রসেস আইসোলেশন (process isolation) এনফোর্স করে, যা নিশ্চিত করে যে সিস্টেমের মেমোরিতে একটি ওয়েবসাইটের রেন্ডারিং কোড অন্যটি থেকে ফিজিক্যালি পার্টিশনড (partitioned) বা আলাদা করা আছে।
- **JIT Compiler ডিজেবল করা:** JavaScript স্পিডের জন্য একটি Just-In-Time (JIT) কম্পাইলারের ওপর নির্ভর করে, কিন্তু JIT ইঞ্জিনগুলো কুখ্যাত রকমের কমপ্লেক্স এবং ব্রাউজারের ভালনারেবিলিটির (vulnerabilities) একটি বিশাল অংশ এদের কারণেই ঘটে। Vanadium ডিফল্টরূপে JIT কম্পাইলার ডিজেবল করে রাখে। তদুপরি, এটি যেসব প্রসেসে JIT ডিজেবল করা আছে, সেগুলোর জন্য ডায়নামিক কোড এক্সিকিউশন সম্পূর্ণরূপে ব্লক করে দেয়। যদি কোনো নির্দিষ্ট ওয়েব অ্যাপ্লিকেশনের ফাংশন করার জন্য এর প্রয়োজন হয়, তবে আপনি ড্রপ-ডাউন মেনুর মাধ্যমে পার-সাইট (per-site) ভিত্তিতে এক্সপ্লিসিটলি (explicitly) JIT কম্পাইলেশনের জন্য অপ্ট-ইন (opt-in) করতে পারেন।
- **Advanced Memory Mitigation:** Vanadium, GrapheneOS-এর `hardened_malloc` (মেমোরি অ্যালোকেটর) এবং হার্ডওয়্যার মেমোরি ট্যাগিং (hardware memory tagging) ব্যবহার করে প্রবাবিলিস্টিক্যালি (probabilistically) ইউজ-আফটার-ফ্রি (use-after-free) বা বাফার ওভারফ্লো (buffer overflow) অ্যাটাকগুলো ডিটেক্ট করে এবং কোনো পেলোড (payload) এক্সিকিউট করার আগেই সেগুলোকে টার্মিনেট করে দেয়।

"কোনো এক্সটেনশন নয়" (No Extensions) রুল: নতুন ইউজাররা প্রায়শই জিজ্ঞাসা করেন কীভাবে Vanadium-এ থার্ড-পার্টি অ্যাড-ব্লকিং (ad-blocking) এক্সটেনশন ইনস্টল করতে হয়। উত্তরটি সোজা: আপনি পারবেন না। Vanadium ইচ্ছাকৃতভাবেই এক্সটেনশন সাপোর্ট করে না। ব্রাউজার এক্সটেনশনগুলো স্ট্রিক্ট সাইট আইসোলেশনকে ব্যাপকভাবে কম্প্রোমাইজ (compromise) করে এবং ওয়েব কনটেন্ট রিড ও মডিফাই করার জন্য ব্রড পারমিশনের (broad permissions) প্রয়োজন হয়। তদুপরি, এক্সটেনশনগুলো আপনার ব্রাউজারকে হাইলি ইউনিক (highly unique) করে তোলে, যার ফলে ব্রাউজার ফিঙ্গারপ্রিন্টিংয়ের (browser fingerprinting) মাধ্যমে আপনাকে সহজেই ট্র্যাক করা যায়। এক্সটেনশনের ওপর নির্ভর করার

পরিবর্তে, Vanadium একটি হাই-পারফরম্যান্স, বিল্ট-ইন কনটেন্ট ফিল্টারিং ইঞ্জিন ইনকর্পোরেট (incorporate) করে যা সিভিয়ার সিকিউরিটি ট্রেডঅফ (severe security tradeoffs) ছাড়াই নেটিভভাবে ম্যালিশিয়াস স্ক্রিপ্ট এবং ট্র্যাকারগুলোকে ব্লক করে দেয়।

GrapheneOS Camera: টেলিমেট্রির চেয়ে প্রাইভেসিকে অগ্রাধিকার দেওয়া

Google এবং Apple দ্বারা প্রোভাইড করা স্ট্যান্ডার্ড ক্যামেরা অ্যাপ্লিকেশনগুলো তাদের নিজ নিজ ক্লাউড ইকোসিস্টেমের সাথে ডিপলি ইন্টিগ্রেটেড (deeply integrated) থাকে। প্রতিবার আপনি যখন কোনো ছবি তোলেন, ক্যামেরা অ্যাপ নীরবে EXIF মেটাডেটা নামক প্রচুর অদেখা টেলিমেট্রি (telemetry) সংযুক্ত করে দেয়। এই মেটাডেটা ছবি তোলার সুনির্দিষ্ট GPS কোঅর্ডিনেট, সময়, তারিখ এবং আপনার ডিভাইসের নির্দিষ্ট হার্ডওয়্যার মডেলকে সরাসরি ইমেজ ফাইলের ভেতরে স্থায়ীভাবে এম্বেড (embed) করে দেয়। আপনি যদি সেই ছবিটি অনলাইনে শেয়ার করেন, তবে আপনি মূলত আপনার ফিজিক্যাল লোকেশন ব্রডকাস্ট (broadcast) করছেন।

অপারেশনাল প্রাইভেসিকে (operational privacy) প্রায়োরিটাইজ বা অগ্রাধিকার দেওয়ার জন্য **GrapheneOS Camera** অ্যাপ্লিকেশনটি একদম গোড়া থেকে তৈরি করা হয়েছে।

- **Metadata Stripping:** ডিফল্টরূপে, এই ক্যামেরা আপনার ছবি এবং ভিডিওগুলো ডিস্কে সেভ করার আগেই সেগুলো থেকে সমস্ত লোকেশন ডেটা এবং ডিভাইস আইডেন্টিফায়ার (identifiers) স্ট্রিপ (strip) বা মুছে ফেলে। আপনি অজান্তেই নিজেকে ডক্সিং (doxxing) করার ভয় ছাড়াই কনফিডেন্টলি আপনার মিডিয়া শেয়ার করতে পারবেন।
- **Integrated QR Scanning:** ক্যামেরায় একটি হাইলি সিকিউর, বিল্ট-ইন QR কোড এবং বারকোড স্ক্যানার অন্তর্ভুক্ত রয়েছে। এটি একটি বড় ধরনের সিকিউরিটি বেনিফিট। স্ট্যান্ডার্ড Android-এ, ইউজাররা প্রায়শই Play Store থেকে থার্ড-পার্টি "QR Scanner" অ্যাপ ডাউনলোড করেন, যেগুলোর মধ্যে অনেকেই হলো অ্যাগ্রেসিভ অ্যাডওয়্যার বা ম্যালওয়ারে লোড করা ট্রোজান (trojans)। GrapheneOS-এর সাথে, নেটিভ ক্যামেরা আনট্রাস্টেড থার্ড-পার্টি সফটওয়্যারের ওপর নির্ভর না করেই সিকিউরভাবে QR কোড হ্যান্ডেল করতে পারে।

GrapheneOS PDF Viewer

ঐতিহাসিকভাবে, PDF হলো ইন্টারনেটের সবচেয়ে বিপজ্জনক ফাইল ফরম্যাটগুলোর মধ্যে অন্যতম। যেহেতু PDF স্ট্যান্ডার্ড এমবেডেড স্ক্রিপ্ট (embedded scripts) এবং কমপ্লেক্স রেন্ডারিং ইনস্ট্রাকশন অ্যালাউ করে, তাই স্টেট-স্পন্সরড অ্যাক্টর এবং সাইবার ক্রিমিনালরা প্রায়শই উইপোনাইজড (weaponized) বা অস্ত্র হিসেবে ব্যবহৃত PDF-গুলো ব্যবহার করে আর্বিট্রারি কোড (arbitrary code) এক্সিকিউট করার জন্য, যখন টার্গেট ব্যক্তি আপাতদৃষ্টিতে নিরীহ মনে হওয়া কোনো ডকুমেন্ট ওপেন করে।

এই বিপজ্জনক ফাইলগুলো পার্স (parse) করার জন্য থার্ড-পার্টি অ্যাপের ওপর নির্ভর করার পরিবর্তে, GrapheneOS-এ একটি প্রোপ্রাইটারি (proprietary) **PDF Viewer** অন্তর্ভুক্ত রয়েছে।

- **Strict Sandboxing:** GrapheneOS PDF Viewer একটি হেভিলি রেস্ট্রিক্টেড স্যান্ডবক্সের (heavily restricted sandbox) ভেতরে কাজ করে। এটি OS-এর এক্সপ্লয়ট মিটিগেশনগুলো (exploit mitigations) লিভারেজ করে এবং Adobe Acrobat-এর মতো কমার্শিয়াল PDF রিডারগুলোতে পাওয়া কমপ্লেক্স, অপ্রয়োজনীয় স্ক্রিপ্টিং ইঞ্জিনগুলো থেকে সম্পূর্ণরূপে মুক্ত।
- **No Network Access:** ভিউয়ারটিকে OS লেভেলে সম্পূর্ণরূপে `Network` পারমিশন ডিনাই (deny) বা প্রত্যাখ্যান করা হয়েছে। এমনকি কোনো হাইলি সফিস্টিকেটেড থ্রেট অ্যাক্টর যদি এমন একটি জিরো-ডে এক্সপ্লয়ট তৈরি করতে সক্ষম হয় যা PDF স্যান্ডবক্সিংকে বাইপাস করে, তবুও পেলোডটিকে ম্যাথমেটিক্যালি আপনার ডেটা এক্সফিলট্রেট (exfiltrate) বা চুরি করার জন্য কোনো কমান্ড-অ্যান্ড-কন্ট্রোল (command-and-control) সার্ভারে "phoning home" থেকে প্রতিরোধ করা হবে।

Apps: সিকিউর সফটওয়্যার রিপোজিটরি

যেহেতু GrapheneOS আউট-অফ-দ্য-বক্সে Google Play Store অন্তর্ভুক্ত করে না, এটি কেবল **Apps** নামক একটি ডেডিকেটেড সিস্টেম অ্যাপ্লিকেশন প্রোভাইড করে। এটি হলো অফিসিয়াল GrapheneOS অ্যাপ রিপোজিটরি ক্লায়েন্ট (repository client)।

`Apps` ক্লায়েন্ট একটি অত্যন্ত স্পেসিফিক, ক্রিটিক্যাল ফাংশন বা কাজ করে: বেইস অপারেটিং সিস্টেম অ্যাপ্লিকেশনগুলোকে সিকিউরলি আপডেট করা। GrapheneOS ফাউন্ডেশন যখন Vanadium, Camera বা PDF Viewer-এর জন্য কোনো আপডেট রিলিজ করে, তখন `Apps` ক্লায়েন্ট সিকিউরভাবে আউট-অফ-ব্যান্ড (out-of-band) পদ্ধতিতে আপডেটটি ফেচ (fetch) করে নিয়ে আসে।

কন্টিনিউয়াস ভেরিফিকেশন (Continuous Verification): স্ট্যান্ডার্ড Android আপডেটের বিপরীতে, GrapheneOS সিস্টেম অ্যাপ আপডেটের জন্য ইনস্টল করার সময় এবং বুট করার সময় উভয় ক্ষেত্রেই স্ট্রিক্ট ক্রিপ্টোগ্রাফিক মেটাডেটা চেক (fs-verity) এনফোর্স করে। `Apps` ক্লায়েন্ট ভেরিফাই করে যে আপডেটটি ফাউন্ডেশনের ট্রাস্টেড কি (trusted key) দিয়ে ম্যাথমেটিক্যালি সাইন (sign) করা আছে কি না, যা কোনো অ্যাডভার্সারিকে ডাউনগ্রেড অ্যাটাক (downgrade attack) (একটি প্যাচ করা অ্যাপকে পুরোনো, ভালনারেবল ভার্সন দিয়ে রিপ্লেস করা) এক্সিকিউট করা থেকে বা একটি প্যাকেজকে কোনো কম্প্রোমাইজড ভ্যারিয়েন্ট (compromised variant) দিয়ে রিপ্লেস করা থেকে প্রিভেন্ট করে।

`Apps` ক্লায়েন্টটি হলো সেই সিকিউর পোর্টাল যার মাধ্যমে আপনি শেষ পর্যন্ত Sandboxed Google Play কম্প্যাটিবিলিটি লেয়ারটি (compatibility layer) ইনস্টল করবেন, যা আমরা পরবর্তী চ্যাপ্টারগুলোতে এক্সটেনসিভলি কভার করব।

ডিফল্ট সফটওয়্যার ইকোসিস্টেমকে এই হার্ডেনড, পারপাস-বিল্ট (purpose-built) টুলগুলোর মধ্যে অ্যাগ্রেসিভলি সীমাবদ্ধ করার মাধ্যমে, GrapheneOS আপনার অ্যাটাক সারফেসকে ড্রাস্টিক্যালি বা ব্যাপকভাবে সংকুচিত করে। আপনি কোনো কম্প্রোমাইজড, ব্লোটেড কমার্শিয়াল প্রোডাক্ট দিয়ে শুরু করছেন না; আপনি একটি স্টেরাইল (sterile), ম্যাথমেটিক্যালি সিকিউর ফাউন্ডেশন বা ভিত্তি দিয়ে শুরু করছেন।

> *[!NOTE] **ভেরিফাইড অ্যাপ ডিরেক্টরি (Verified App Directory)***
> *কোন থার্ড-পার্টি অ্যাপগুলো নিরাপদ এবং কোনগুলো নজরদারি করে, তা জানতে আমাদের অফিসিয়াল Graphene OS অ্যাপ্লিকেশন ডিরেক্টরি ভিজিট করুন। এখানে ১০০% ভেরিফাইড FOSS অ্যাপের তালিকা এবং বিপজ্জনক টেলিমেট্রি অ্যাপের ডেটাবেজ দেওয়া আছে।*

১.৯ দ্য মাইন্ডসেট: বেসিক প্রাইভেসি হ্যাবিটস (Operational Security 101)

আপনি সফলভাবে সার্ভেইল্যান্স ইঞ্জিন রিপ্লেস করেছেন। আপনি Bootloader লক করেছেন, ক্রিপ্টোগ্রাফিক কি (cryptographic keys) এনরোল করেছেন, ব্যাকগ্রাউন্ড রেডিও সাইলেন্স করেছেন এবং আপনার অথেন্টিকেশন প্রোটোকল হার্ডেন (harden) করেছেন। ডিজিটাল কংক্রিট এখন পুরোপুরি জমে গেছে। আপনি এখন একটি ম্যাথমেটিক্যালি ভেরিফাইড ফোর্ট্রেস (mathematically verified fortress) এর ভেতর অবস্থান করছেন।

তবে একটি ফোর্ট্রেস বা দুর্গ সম্পূর্ণ অর্থহীন, যদি এর অপারেটর ক্রমাগত সামনের দরজা হাট করে খোলা রাখে।

আর এটিই আমাদের ডিজিটাল সভারেনটি (digital sovereignty)-এর সবচেয়ে ক্রিটিক্যাল, অথচ প্রায়শই অবহেলিত একটি কম্পোনেন্টের দিকে নিয়ে যায়: **Operational Security (OpSec)**। GrapheneOS এই গ্রহের সবচেয়ে অ্যাডভান্সড, সিকিউর মোবাইল অপারেটিং সিস্টেম, কিন্তু দিনশেষে এটি কেবলই একটি টুল। এটি একটি শিল্ড বা ঢাল। এটি আপনাকে আপনার নিজের হাত থেকে রক্ষা করতে পারবে না। আপনি যদি একটি এনক্রিপ্টেড অপারেটিং সিস্টেম ইন্সটল করার পরপরই Facebook-এ লগ ইন করেন, আপনার দৈনন্দিন যাতায়াতের ছবি আপলোড করেন এবং একটি ওয়েদার অ্যাপকে লোকেশন পারমিশন দেন, তবে আপনি হার্ডওয়্যারের মূল উদ্দেশ্যটাই ধ্বংস করে দিচ্ছেন। আপনি যদি স্বেচ্ছায় ডেটা তুলে দেন, তবে অ্যাডভার্সারি (adversary)-এর আপনার এনক্রিপশন ক্র্যাক করার কোনো প্রয়োজনই নেই।

প্রকৃত প্রাইভেসির জন্য আচরণগত একটি ফান্ডামেন্টাল শিফট বা মৌলিক পরিবর্তন প্রয়োজন। এর জন্য একটি OpSec মাইন্ডসেট অ্যাডপ্ট করা জরুরি। যেহেতু আমরা আমাদের ফাউন্ডেশনাল সেটআপ শেষ করছি, আপনার হার্ডওয়্যারের শক্তির সাথে আপনার প্রতিটি কাজের সামঞ্জস্য নিশ্চিত করতে এখন থেকেই আপনাকে এই বেসিক প্রাইভেসি হ্যাবিটগুলো ইমপ্লিমেন্ট করা শুরু করতে হবে।

১. সার্চ ইঞ্জিন আম্বিলিক্যাল কর্ড ছিন্ন করুন (Sever the Search Engine Umbilical Cord)

আজই আপনি যে সবচেয়ে সহজ এবং হাই-ইমপ্যাক্ট পরিবর্তনটি করতে পারেন, তা হলো এই প্যানোপটিকন (panopticon)-কে আপনার দৈনন্দিন চিন্তাভাবনা ফিড করানো বন্ধ করা। Google Search কেবল তথ্য খোঁজার কোনো টুল নয়; এটি মানব ইতিহাসের সবচেয়ে সোফিস্টিকেটেড সাইকোলজিক্যাল প্রোফাইলিং ইঞ্জিন। আপনি যে রোগের লক্ষণ সার্চ করেন, যে পলিটিক্যাল কোয়েরি টাইপ করেন এবং যে প্রোডাক্ট নিয়ে রিসার্চ করেন—তার প্রতিটি লগ করা হয়, কোরিলেট করা হয় এবং স্থায়ীভাবে আপনার শ্যাডো প্রোফাইল (shadow profile)-এর সাথে যুক্ত করে দেওয়া হয়।

The Fix: অবিলম্বে আপনার ডিফল্ট সার্চ ইঞ্জিন পরিবর্তন করুন। Vanadium ওপেন করুন, **Settings > Search engine**-এ ন্যাভিগেট করুন এবং একটি প্রাইভেসি-রেস্পেক্টিং অল্টারনেটিভ নির্বাচন করুন।

- **DuckDuckGo:** এটি একটি সলিড, বিগিনার-ফ্রেন্ডলি অল্টারনেটিভ যা আপনার IP অ্যাড্রেস ট্র্যাক করে না বা আপনার সার্চ হিস্ট্রি লগ করে না।
- **Brave Search:** এটি সম্পূর্ণ স্ক্র্যাচ থেকে তৈরি একটি ইন্ডিপেনডেন্ট ইনডেক্স, যা Google-এর অ্যালগরিদম থেকে পুরোপুরি মুক্ত এবং কোনো ট্র্যাকিং টেলিমেট্রি (tracking telemetry) ছাড়াই এক্সেলেন্ট রেজাল্ট অফার করে।

২. আপনার কমিউনিকেশন মাইগ্রেট করুন (The Death of SMS)

স্ট্যান্ডার্ড SMS টেক্সট মেসেজিং ফান্ডামেন্টালি ব্রোকেন (fundamentally broken)। এটি একটি সেকেলে, আনএনক্রিপ্টেড প্রোটোকল যা প্লেইন টেক্সট (plain text)-এ আপনার টেলিকম ক্যারিয়ারের সার্ভারের মধ্য দিয়ে মেসেজ রাউট করে। ক্যারিয়ার লগ করে যে আপনি কাকে টেক্সট করেন, কখন করেন এবং মেসেজে ঠিক কী বলা হয়েছে। তাছাড়া, WhatsApp-এর মতো প্ল্যাটফর্মগুলো—মেসেজ কনটেন্টের জন্য এন্ড-টু-এন্ড এনক্রিপশন (end-to-end encryption) ব্যবহার করা সত্ত্বেও—মূলত Meta-র মালিকানাধীন। তারা অত্যন্ত আগ্রাসীভাবে মেটাডেটা (আপনি কার সাথে কথা বলেন, কতবার বলেন এবং কোন IP অ্যাড্রেস থেকে বলেন) হারভেস্ট এবং মনিটাইজ করে।

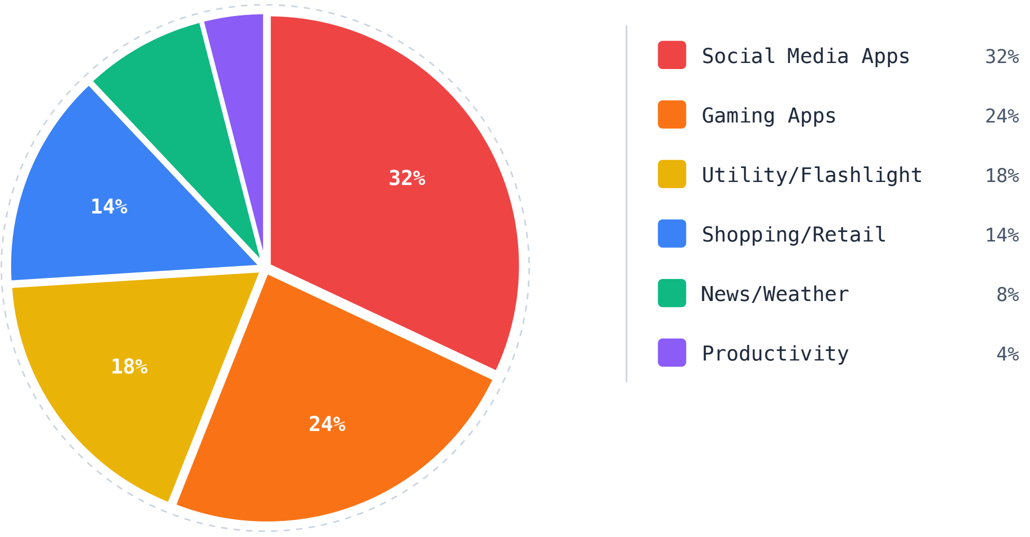
The Fix: আপনার কমিউনিকেশন **Signal**-এ ট্রানজিশন করুন। Signal হচ্ছে সিকিউর, এনক্রিপ্টেড কমিউনিকেশনের জন্য গোল্ড স্ট্যান্ডার্ড। এটি ওপেন-সোর্স, ইন্ডিপেনডেন্টলি অডিটেড এবং একটি নন-প্রফিট ফাউন্ডেশন দ্বারা পরিচালিত। আপনার মেসেজ এবং কলগুলোর কনটেন্ট যে শুধু ক্রিপ্টোগ্রাফিক্যালি শিল্ডেড (cryptographically shielded) থাকে তা-ই নয়, বরং মেটাডেটা লুকানোর জন্য Signal "Sealed Sender" নামক একটি টেকনোলজিও ব্যবহার করে। আপনি কাকে মেসেজ পাঠাচ্ছেন তা Signal নিজেও জানে না, এবং এটি বের করার জন্য তাদের কাছে কোনো কি (keys) নেই। আপনার পরিবার, বন্ধুবান্ধব এবং বিজনেস অ্যাসোসিয়েটদের Signal-এ মুভ করার একটি অভ্যাস গড়ে তুলুন।

> 💡 ***Pro Tip:*** *SMS-কে একটি কম্প্রোমাইজড চ্যানেল হিসেবে বিবেচনা করুন। SMS-এর মাধ্যমে পাঠানো যেকোনো মেসেজ আপনার ক্যারিয়ার, রেঞ্জের মধ্যে থাকা যেকোনো Stingray এবং SS7 নেটওয়ার্ক অ্যাক্সেস রয়েছে এমন যেকোনো এনটিটির কাছে দৃশ্যমান। সমস্ত পার্সোনাল এবং প্রফেশনাল কমিউনিকেশনের জন্য Signal-কে ডিফল্ট হিসেবে সেট করুন।*

৩. আপনার ইমেইল De-Google করুন

আপনি যদি এখনও একটি `@gmail.com` বা `@yahoo.com` ইমেইল অ্যাড্রেস ব্যবহার করে থাকেন, তবে ধরে নিন আপনার ইনবক্স অ্যালগরিদমিক্যালি স্ক্যান করা হচ্ছে। আপনার জীবনের একটি নিখুঁত অ্যাডভার্টাইজিং প্রোফাইল তৈরির জন্য আপনার রিসিভ করা প্রতিটি রসিদ, আপনার বুক করা প্রতিটি ফ্লাইট এবং প্রতিটি ব্যক্তিগত কথোপকথন ইনজেস্ট (ingest) করা হচ্ছে। ইমেইল হলো আপনার ডিজিটাল পাসপোর্ট; এটি আপনার ব্যাংক অ্যাকাউন্ট, আপনার মেডিক্যাল রেকর্ড এবং আপনার আইডেন্টিটির সাথে ওতপ্রোতভাবে যুক্ত। এটিকে অবশ্যই সুরক্ষিত করতে হবে।

UNNECESSARY PERMISSION REQUESTS BY APP CATEGORY



Source: Analysis based on Exodus Privacy, ICSI Netalyzr, and AppCensus audit data

The Fix: একটি প্রাইভেসি-রেস্পেক্টিং জুরিসডিকশন (privacy-respecting jurisdiction) ভিত্তিক এন্ড-টু-এন্ড এনক্রিপ্টেড ইমেইল প্রোভাইডারে ট্রানজিশন শুরু করুন।

- **ProtonMail:** সুইজারল্যান্ড-ভিত্তিক Proton জিরো-অ্যাক্সেস এনক্রিপশন (zero-access encryption) প্রদান করে। এমনকি যদি কোনো কোর্ট অর্ডারও দেওয়া হয়, তবুও Proton ম্যাথমেটিক্যালি আপনার ইনবক্সের কনটেন্ট পড়তে পারবে না, কারণ তাদের কাছে আপনার প্রাইভেট ডিক্রিপশন কি (private decryption keys) নেই।
- **Tuta (formerly Tutanota):** জার্মানি-ভিত্তিক Tuta কেবল মেসেজ বডি-ই নয়, বরং সাবজেক্ট লাইন এবং আপনার সম্পূর্ণ কন্টাক্ট বুককেও এনক্রিপ্ট করে।

ইমেইল মাইগ্রেট করতে সময় লাগে। আপনার সবচেয়ে ক্রিটিক্যাল অ্যাকাউন্টগুলো (ব্যাংকিং, হেলথকেয়ার, প্রাইমারি কমিউনিকেশন) আপনার নতুন এনক্রিপ্টেড অ্যাড্রেসে মুভ করার মাধ্যমে শুরু করুন এবং ধীরে ধীরে লিগ্যাসি অ্যাকাউন্টটিকে ফেজ আউট (phase out) করে দিন।

৪. রুথলেস পারমিশন ম্যানেজমেন্ট: দ্য লোকেশন লাই (Ruthless Permission Management: The Location Lie)

আধুনিক স্মার্টফোন যুগের অন্যতম বড় প্রতারণা হলো "Location" পারমিশনের নরমালাইজেশন। আপনার শহরে বৃষ্টি হচ্ছে কি না তা জানানোর জন্য একটি ওয়েদার অ্যাপের আপনার প্রিসাইজ GPS কোঅর্ডিনেটস (precise GPS coordinates) জানার কোনো প্রয়োজন নেই। আপনি কোন এলাকায় আছেন তা একটি ফ্ল্যাশলাইট অ্যাপের জানার দরকার নেই। ফাস্ট-ফুড রেস্তোরাঁর অ্যাপগুলো আপনার লোকেশন ডিমান্ড করে আপনাকে বার্গার খুঁজে পেতে সাহায্য করার জন্য নয়, বরং আপনার দৈনন্দিন ট্রাভেল রুটিন ম্যাপ করার জন্য এবং সেই মুভমেন্ট ডেটা থার্ড-পার্টি ডেটা ব্রোকারদের কাছে বিক্রি করার জন্য।

The Fix: ডিফল্টভাবেই ডিনাই (Deny) করুন। যখন আপনি অ্যাপ্লিকেশন ইন্সটল করা শুরু করবেন, আপনাকে অবশ্যই রুথলেস হতে হবে:

- ৯৯% অ্যাপ্লিকেশনের কাজ করার জন্য আপনার লোকেশন, মাইক্রোফোন বা ক্যামেরার কোনো প্রয়োজন নেই।
- যদি কোনো অ্যাপ এমন কোনো পারমিশন রিকোয়েস্ট করে যা তার নিতান্তই প্রয়োজন নেই, তবে তা ডিনাই করুন।
- যদি অ্যাপটি সেই পারমিশন ছাড়া ফাংশন করতে অস্বীকৃতি জানায়, তবে অ্যাপটি আনইন্সটল করে দিন।

আপনার লোকেশন ডেটা হলো অত্যন্ত সেনসিটিভ ইন্টেলিজেন্স। এটি প্রকাশ করে দেয় আপনি কোথায় ঘুমান, কোথায় কাজ করেন, কাদের সাথে মেলামেশা করেন এবং কোন কোন মেডিক্যাল ক্লিনিকে যান। এটিকে সর্বোচ্চ শক্তিমত্তার সাথে প্রটেক্ট করুন।

৫. সোশ্যাল ওভারশেয়ারিং এর বিপদ (The Danger of Social Oversharing)

সবশেষে, আমাদের অবশ্যই হিউম্যান এলিমেন্ট (human element) নিয়ে কথা বলতে হবে। আপনি পাবলিকলি কী পোস্ট করছেন তা দেখে তথ্য সংগ্রহ করাটা, বিশ্বের সবচেয়ে সোফিস্টিকেটেড এক্সপ্লয়েট (exploit) চালানোর চেয়ে বহুগুণ কম ব্যয়বহুল এবং কম সময়সাপেক্ষ।

অ্যাডভার্সারিরা—তা সে কর্পোরেট ডেটা ব্রোকার, স্টকার (stalker) বা হোস্টাইল ইন্টেলিজেন্স (hostile intelligence) যাই হোক না কেন—ওপেন সোর্স ইন্টেলিজেন্স (OSINT)-এর উপর ব্যাপকভাবে নির্ভর করে। আপনি যদি আপনার সকালের কফির একটি ছবি পোস্ট করেন, তবে আপনি প্রকাশ করে দিচ্ছেন আপনি কোন ক্যাফেতে নিয়মিত যাতায়াত করেন। আপনি যদি কর্মক্ষেত্রে কোনো হতাশাজনক দিন সম্পর্কে পোস্ট করেন, তবে আপনি আপনার এমপ্লয়ার এবং আপনার ইমোশনাল স্টেট প্রকাশ করে দিচ্ছেন। আপনার প্রতিটি পোস্টই একেকটি পাজল পিস (puzzle piece)।

আপনাকে অবশ্যই অপারেশনাল ডিসিপ্লিন (operational discipline) কাল্টিভেট করতে হবে। পোস্ট করার আগে চিন্তা করুন। আপনার ছবির ব্যাকগ্রাউন্ড ব্লার করুন, মেটাডেটা স্ট্রিপ করুন (যা GrapheneOS Camera স্বয়ংক্রিয়ভাবেই হ্যান্ডেল করে), এবং নিজেকে প্রশ্ন করুন: *এই ইনফরমেশনটি কি আমার বিরুদ্ধে ওয়েপোনাইজ (weaponize) করা সম্ভব?*

৬. আর্টিফিশিয়াল ইন্টেলিজেন্স (AI) এবং আপনার প্রাইভেসি

বর্তমান যুগে প্রোডাক্টিভিটির জন্য AI (যেমন: ChatGPT, Gemini, Claude) প্রায় অপরিহার্য হয়ে উঠলেও, একজন অপসেক (OpSec) প্র্যাকটিশনারের দৃষ্টিকোণ থেকে কমার্শিয়াল AI-গুলো হলো ডেটা হারভেস্টিংয়ের চূড়ান্ত হাতিয়ার। আপনি যখন এই AI গুলোর অফিশিয়াল মোবাইল অ্যাপ ইনস্টল করেন, তখন তারা ব্যাকগ্রাউন্ডে প্রচুর টেলিমেট্রি (ডিভাইস আইডি, আইপি অ্যাড্রেস) কালেক্ট করে এবং আপনার দেওয়া প্রতিটি প্রম্পট তাদের সার্ভারে ভবিষ্যতের ট্রেইনিংয়ের জন্য সেভ করে রাখে। আপনার প্রশ্নগুলো বিশ্লেষণ করে তারা আপনার একটি নিখুঁত সাইকোলজিক্যাল প্রোফাইল তৈরি করতে পারে।

The Fix: কম্পার্টমেন্টালাইজেশন (Compartmentalization) এবং Local AI।

- **অ্যাপ বর্জন করুন:** GrapheneOS-এ কখনোই ChatGPT, Gemini বা Claude-এর অফিশিয়াল অ্যাপ ইনস্টল করবেন না।
- **ব্রাউজার আইসোলেশন:** যদি কমার্শিয়াল AI ব্যবহার করতেই হয়, তবে Vanadium ব্রাউজার দিয়ে ওয়েব ভার্সন (web version) ব্যবহার করুন। আসল নাম বা ইমেইল দিয়ে লগইন করবেন না এবং PII (Personally Identifiable Information) ইনপুট দেওয়া থেকে বিরত থাকুন।
- **Local AI (The OpSec Holy Grail):** যদি আপনার থ্রেট মডেল অত্যন্ত হাই (High) হয়, তবে পিসিতে লোকাল এআই (যেমন: LM Studio, Ollama) ব্যবহার করা শুরু করুন। লোকাল এআই সম্পূর্ণ অফলাইনে রান করে, যার ফলে জিরো টেলিমেট্রি এবং জিরো ডেটা হারভেস্টিং নিশ্চিত হয়।

কুইক-রেফারেন্স: প্রাইভেসি টুলকিট (Quick-Reference: Privacy Toolkit)

নিচের টেবিলে এই সম্পূর্ণ চ্যাপ্টারে রিকমেন্ড করা এসেনশিয়াল টুল এবং সার্ভিসগুলো সামারাইজ করা হয়েছে। এর প্রতিটিকে ক্রিপ্টোগ্রাফিক ট্রান্সপারেন্সি, ওপেন-সোর্স অডিটিবিলিটি এবং মিনিমাল মেটাডেটা কালেকশনের জন্য ভেটিং (vetting) করা হয়েছে।

ক্যাটেগরি	রিকমেন্ডেড টুল	জুরিসডিকশন	কি প্রাইভেসি ফিচার
Search Engine	DuckDuckGo	USA	কোনো IP/query লগিং নেই
Search Engine	Brave Search	USA	ইন্ডিপেনডেন্ট ইনডেক্স; কোনো ট্র্যাকিং নেই
Messaging	Signal	USA (Non-profit)	E2E এনক্রিপশন + Sealed Sender মেটাডেটা প্রটেকশন
Email	ProtonMail	Switzerland	জিরো-অ্যাক্সেস এনক্রিপশন; আপনার মেইল পড়তে পারে পণ্ডিত নয়
Email	Tuta	Germany	সাবজেক্ট লাইন + কন্টাক্ট বুক এনক্রিপ্ট করে
Browser	Vanadium	GrapheneOS built-in	হার্ডেনড Chromium; ডিফল্টরূপে JIT ডিজেবলড
VPN	Mullvad	Sweden	নো-অ্যাকাউন্ট, ক্যাশ-পেয়েবল; কোনো লগিং নেই

আপনি **পার্ট ১: দ্য ফাউন্ডেশন (Part I: The Foundation)** সম্পন্ন করেছেন। আপনি সফলভাবে আপনার হার্ডওয়্যার রিক্লেইম করেছেন, একটি ম্যাথমেটিক্যালি ভেরিফাইড বুট প্রসেস এস্টাবলিশ করেছেন এবং প্রকৃত অপারেশনাল সিকিউরিটির জন্য প্রয়োজনীয় সাইকোলজিক্যাল শিফটিং শুরু করেছেন। আপনার ডিভাইসটি এখন একটি স্টেরাইল (sterile), হার্ডেনড এনভায়রনমেন্ট।

তবে একটি স্টেরাইল এনভায়রনমেন্ট কেবল একটি শুরু মাত্র। আধুনিক বিশ্বে ফাংশন করার জন্য, একটি স্মার্টফোনকে তার হোস্টকে কম্প্রোমাইজ না করেই কানেক্ট, কমিউনিকেট এবং কমপ্লেক্স অ্যাপ্লিকেশন এক্সিকিউট করতে হয়। এটি এখন দেয়ালগুলো রিইনফোর্স (reinforce) করার, ইন্টারনাল নেটওয়ার্ক আর্কিটেক্ট করার এবং থ্রেটগুলোকে আইসোলেট (isolate) করার সময়। পাতা উল্টান, এবং চলুন গড়ে তুলি **পার্ট ২: দ্য ফোর্ট্রেস (Part II: The Fortress)**।

Part II: The Fortress

চ্যাপ্টার ২ ভূমিকা (Chapter 2 Introduction)

পার্ট ১-এ আপনি ভিত্তিপ্রস্তর স্থাপন করেছেন। ম্যানুফ্যাকচারার কর্তৃক ইনস্টল করা নেটিভ সার্ভেইল্যান্স ইঞ্জিনটি আপনি সফলভাবে নির্মূল করেছেন, হার্ডওয়্যারকে ক্রিপ্টোগ্রাফিক্যালি সিল (cryptographically seal) করেছেন এবং একটি ম্যাথমেটিক্যালি ভেরিফাইড বেইসলাইন (mathematically verified baseline) প্রতিষ্ঠা করেছেন। আপনার ডিভাইসটি এখন আর কোনো ট্র্যাকিং বিকন (tracking beacon) নয়; এটি এখন একটি ব্ল্যাংক স্লেট (blank slate)। আপনি সাধারণ কনজ্যুমারদের জগত থেকে উত্তীর্ণ হয়ে অ্যাবসলিউট ডিজিটাল সভারেনটি (absolute digital sovereignty) বা পূর্ণাঙ্গ ডিজিটাল সার্বভৌমত্বের ডোমেইনে পা রেখেছেন।

তবে একটি ব্ল্যাংক স্লেট বা শূন্য ক্যানভাস কোনো ফাংশনিং ওয়ার্কস্টেশন হতে পারে না।

আধুনিক বিশ্বে কার্যকরভাবে অপারেট করার জন্য, আপনার ডিভাইসটিকে শেষ পর্যন্ত নেটওয়ার্কের সাথে কানেক্ট করতে হবে, অ্যাপ্লিকেশন ডাউনলোড করতে হবে এবং বাইরের পরিবেশের সাথে ইন্টার‍্যাক্ট করতে হবে। আপনি যে অ্যাপ্লিকেশনটিই ইনস্টল করেন, যে নেটওয়ার্কেরই সাথে কানেক্ট করেন এবং যে ফাইলটিই ওপেন করেন—তার প্রতিটিই আপনার জন্য রিস্ক বা ঝুঁকি নিয়ে আসে। এই বইয়ের পার্ট ২, **দ্য ফোর্ট্রেস (The Fortress)**, আপনার অপারেটিং সিস্টেমের ইন্টারনাল ডিফেন্স বা অভ্যন্তরীণ প্রতিরক্ষা ব্যবস্থা আর্কিটেক্ট করার জন্য ডেডিকেটেড। আমরা আর ডিফল্ট সেটিংসের ওপর নির্ভর করব না। এর বদলে আমরা অ্যাক্টিভলি সেগমেন্টেড ভল্ট (segmented vaults) তৈরি করব, এনক্রিপ্টেড টানেলের মাধ্যমে ট্রাফিক রাউট করব এবং হোস্টাইল কোডগুলোকে (hostile code) স্যান্ডবক্সিং (sandboxing) করব।

তবে, আর্কিটেকচার নির্মাণের কাজ শুরু করার আগে আমাদের অবশ্যই আপনার অপারেশনাল মাইন্ডসেটকে এলিভেট (elevate) করতে হবে। পার্ট ১-এর শেষে আলোচিত বেসিক প্রাইভেসি হ্যাবিটগুলো—যেমন সিকিউর সার্চ ইঞ্জিন ব্যবহার না করা এবং SMS বর্জন না করা—যেমন সিকিউর সার্চ ইঞ্জিন ব্যবহার না করা এবং SMS বর্জন না করা—হলো রাতে সদর দরজা লক না করে ঘুমানোর সমতুল্য। এটি দ্রুত চেঞ্জ করা প্রয়োজন, শুধু তাই নয়, এগুলো মানার পর, আমরা এরপর যে থ্রেটগুলোর (threats) মোকাবেলা করতে যাচ্ছি তার জন্য এটি পর্যাপ্ত নয়।

আপনার ডিজিটাল জীবনের প্রতি একটি ট্যাকটিক্যাল (tactical) এবং আনকম্প্রোমাইজিং (uncompromising) অ্যাপ্রোচ অ্যাডপ্ট করার এখনই সময়।

২.০ দ্য অ্যাডভান্সড অপসেক মাইন্ডসেট (The Advanced OpSec Mindset)

অপারেশনাল সিকিউরিটি (Operational Security), বা সংক্ষেপে OpSec, হলো একটি মিলিটারি ডকট্রিন (military doctrine)। এটি মূলত ক্রিটিক্যাল ইনফরমেশন আইডেন্টিফাই করা, অ্যাডভার্সারিরা (adversaries) কীভাবে সেই ইনফরমেশন এক্সট্রাক্ট করতে পারে তা নির্ধারণ করা এবং সেই ভালনারেবিলিটিগুলো (vulnerabilities) নির্মূল করার জন্য কাউন্টারমেজার (countermeasures) এক্সিকিউট করার একটি অ্যানালিটিক্যাল প্রসেস।

GrapheneOS-এ অ্যাডভান্সড OpSec-এর অর্থ হলো, এই রিয়েলাইজেশনটা থাকা যে আপনার স্মার্টফোনটি শুধু কোনো ফোন নয়। এটি হলো আপনার জীবনের যাবতীয় ইন্টেলিজেন্সের (intelligence) সবচেয়ে কনসেন্ট্রেটেড রিপোজিটরি (concentrated repository)। এটি আপনার বায়োমেট্রিক্স, আপনার ফাইন্যান্সিয়াল কি (financial keys), আপনার ফিজিক্যাল মুভমেন্ট এবং আপনার প্রাইভেট কথোপকথন ধারণ করে।

প্রকৃত প্রাইভেসি কোনো সফটওয়্যার নয়। সফটওয়্যার হলো স্ট্যাটিক। প্রকৃত প্রাইভেসি হলো একটি রিলেন্টলেস (relentless) এবং কন্টিনিউয়াস (continuous) আচরণ। GrapheneOS আপনাকে মিলিটারি-গ্রেড টুল সরবরাহ করে, কিন্তু আপনি হলেন এর অপারেটর। আর একজন অপারেটরকে অবশ্যই ডিসিপ্লিন বা শৃঙ্খলার সাথে কাজ করতে হবে।

দ্য আপগ্রেডেড থ্রেট মডেল (The Upgraded Threat Model)

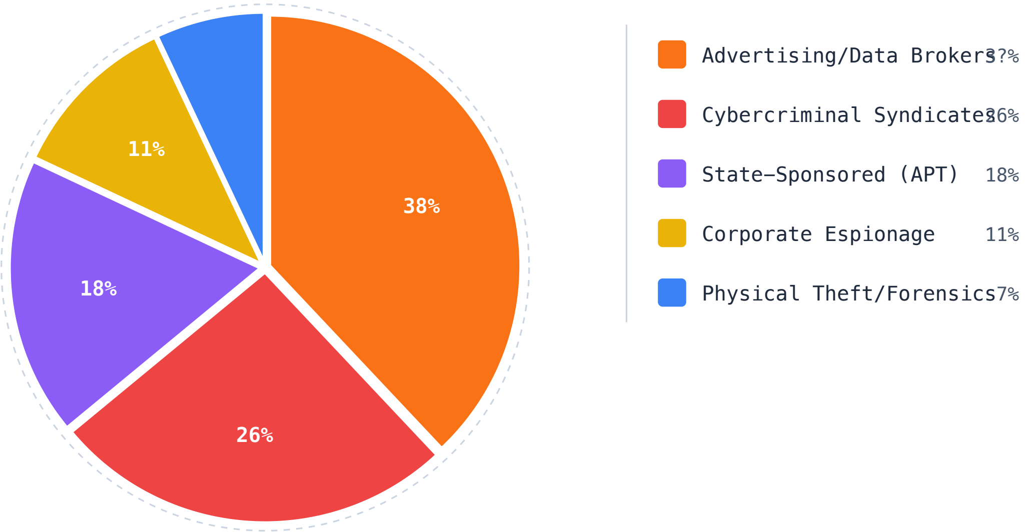
অ্যাডভান্সড OpSec-এর প্রয়োজনীয়তা বোঝার জন্য আমাদের অবশ্যই আমাদের থ্রেট মডেলটিকে (threat model) রিডিফাইন করতে হবে। পার্ট ১-এ, আমাদের প্রধান অ্যাডভার্সারি ছিল বিভিন্ন অ্যাডভার্টাইজিং কংগ্লোমারেট (advertising conglomerates) এবং ডেটা ব্রোকার—যারা টার্গেটেড অ্যাডভার্টাইজমেন্ট বিক্রি করার জন্য প্যাসিভলি ডেটা হারভেস্ট করছিল।

পার্ট ২-এ, আমরা আমাদের ডিফেন্সিভ পোসচার (defensive posture) এস্কেলেট (escalate) করব। আমরা এখন এমন একটি সিস্টেম আর্কিটেক্ট করছি যা যেকোনো অ্যাক্টিভ এবং টার্গেটেড অ্যাগ্রেশন (targeted aggression) রুখে দেওয়ার জন্য ডিজাইন করা হয়েছে। আমাদের বর্তমান থ্রেট মডেলের অন্তর্ভুক্ত হলো:

১. কর্পোরেট এসপিয়নাজ এবং স্টকারস (Corporate Espionage & Stalkers): হাইলি মোটিভেটেড ব্যক্তি বা রাইভাল এনটিটি (rival entities) যারা অ্যাক্টিভলি আপনার কমিউনিকেশন কম্প্রোমাইজ করতে, ইন্টেলেকচুয়াল প্রোপার্টি (intellectual property) এক্সট্রাক্ট করতে বা ফিজিক্যালি আপনাকে ট্র্যাক করতে চেষ্টা করছে। **২. টার্গেটেড ডিজিটাল সার্ভেইল্যান্স (Targeted Digital Surveillance):** স্টেট-স্পন্সরড অ্যাক্টর, ইন্টেলিজেন্স এজেন্সি বা অ্যাডভান্সড সাইবার

ক্রিমিনাল সিন্ডিকেট, যারা ম্যালিশিয়াস লিংক, ওয়েপোনাইজড PDF বা কম্প্রোমাইজড অ্যাপ্লিকেশনের মাধ্যমে জিরো-ডে রিমোট কোড এক্সিকিউশন (zero-day RCE) এক্সপ্লয়েট ডিপ্লয় করে। ৩. **ফিজিক্যাল ফরেনসিক সিজারস (Physical Forensic Seizures):** ফিজিক্যালি আপনার হার্ডওয়্যার বাজেয়াপ্ত বা চুরি হয়ে যাওয়া, যার ফলশ্রুতিতে সোফিস্টিকেটেড ফরেনসিক এক্সট্রাকশন মেশিন (যেমন Cellebrite বা GrayKey) প্রয়োগ করে আপনার এনক্রিপশন ব্রুট-ফোর্স (brute-force) করার চেষ্টা করা এবং সিলিকন চিপ থেকে সরাসরি ডেটা রিপ (rip) করে নেওয়া।

MOBILE THREAT ACTOR CLASSIFICATION (2025)



Source: Analysis derived from Mandiant M-Trends, CrowdStrike Global Threat Report, and ENISA Threat Landscape

যখন আপনি বুঝতে পারবেন যে আপনি *এই ধরনের* অ্যাডভার্সারিদের বিরুদ্ধে নিজেকে ডিফেন্ড করছেন, তখন আপনার আচরণ বদলে যাবে। আপনি আর কনভিনিয়েন্সকে (convenience) কোনো ফিচার হিসেবে দেখবেন না; আপনি কনভিনিয়েন্সকে দেখবেন একটি ভালনারেবিলিটি (vulnerability) হিসেবে।

কোর ফিলোসফি: অ্যাটাক সারফেস রিডাকশন (The Core Philosophy: Attack Surface Reduction)

অ্যাডভান্সড OpSec-এর সেন্ট্রাল পিলার এবং স্বয়ং GrapheneOS-এর পেছনের কোর ইঞ্জিনিয়ারিং ফিলোসফি হলো **Attack Surface Reduction (অ্যাটাক সারফেস রিডাকশন)**।

আপনার "অ্যাটাক সারফেস" হলো সেই সমস্ত ভিন্ন ভিন্ন পয়েন্টের (যাকে "অ্যাটাক ভেক্টর" বলা হয়) সমষ্টি, যেখান দিয়ে একজন আনঅথোরাইজড ইউজার আপনার এনভায়রনমেন্টে ডেটা এন্টার করাতে বা তা থেকে ডেটা এক্সট্রাক্ট করার চেষ্টা করতে পারে। কোডের প্রতিটি লাইন, প্রতিটি অ্যাক্টিভ রেডিও ফ্রিকোয়েন্সি, প্রতিটি ব্যাকগ্রাউন্ড সার্ভিস এবং প্রতিটি ইনস্টল করা অ্যাপ্লিকেশন আপনার অ্যাটাক সারফেসের আকার বাড়িয়ে দেয়।

কোনো ফোর্ট্রেস বা দুর্গ রক্ষা করার জন্য আপনি একশোটি দরজা বানিয়ে সেগুলোর সবকটিতে তালা লাগানোর চেষ্টা করেন না। আপনি কেবল একটি দরজাই বানান এবং বাকি দেয়ালটিকে সলিড কংক্রিট দিয়ে সিল করে দেন।

বাস্তবে, অ্যাটাক সারফেস রিডাকশনের জন্য এক্সট্রিম মিনিমালিজম (extreme minimalism) এবং রুথলেস ডিসিপ্লিনের (ruthless discipline) প্রয়োজন হয়। যদি কোনো ফিচার অ্যাক্টিভলি ব্যবহৃত না হয়, তবে এটিকে অবশ্যই ডিজেবল করে রাখতে হবে। যদি কোনো অ্যাপ্লিকেশন আপনার অপারেশনের জন্য অপরিহার্য না হয়, তবে এটিকে কোনোভাবেই ইনস্টল করা যাবে না।

আপনার হার্ডওয়্যার রেডিওগুলোর কথাই বিবেচনা করুন। NFC, Bluetooth এবং Ultra-Wideband হলো দারুণ কিছু কনভিনিয়েন্স। কিন্তু যদি এগুলোকে সবসময় অন করে রাখা হয়, তবে সেগুলো হয়ে দাঁড়ায় স্থায়ীভাবে উন্মুক্ত লিসেনিং পোর্ট (listening ports)। একজন সোফিস্টিকেটেড অ্যাডভার্সারি কোনো Bluetooth ভালনারেবিলিটি লিভারেজ করে পঞ্চাশ ফুট দূর থেকেই আপনার ডিভাইস কম্প্রোমাইজ করে ফেলতে পারে—এমনকি আপনার পকেট থেকে ফোন বের করার আগেই।

একজন এলিট অপারেটর কখনোই তাদের Bluetooth অন করে ফেলে রাখে না। তারা এটি অন করে, তাদের ডিভাইস কানেক্ট করে, টাস্ক শেষ করে এবং সাথে সাথেই সেটি অফ করে দেয়। অপারেশন শেষ হওয়ার মুহূর্তেই তারা তাদের অ্যাটাক সারফেস শূন্যে নামিয়ে আনে।

এই ফিলোসফি হার্ডওয়্যার রেডিওর সীমানা ছাড়িয়ে আরও বহুদূর পর্যন্ত বিস্তৃত। এটি ডিক্টেট (dictate) করে কীভাবে GrapheneOS তার কোর ইনফ্রাস্ট্রাকচার ম্যানেজ করে:

- **USB-C Port Control:** ডিফল্টভাবেই, আপনার ডিভাইসটি লক থাকা অবস্থায় GrapheneOS আপনার USB-C পোর্টের ডেটা লাইনগুলোকে ফিজিক্যালি ডিজেবল করে দেয়। এটি ম্যালিশিয়াস চার্জিং ক্যাবল বা ফরেনসিক এক্সট্রাকশন ডিভাইস ব্যবহার করে চালানো যেকোনো ফিজিক্যাল অ্যাটাককে মুহূর্তের মধ্যেই নিউট্রালাইজ (neutralize) করে দেয়।
- **Disabling Dynamic Code:** GrapheneOS অত্যন্ত অ্যাগ্রেসিভলি সম্পূর্ণ অপারেটিং সিস্টেম জুড়ে ডায়নামিক, ইন-মেমরি কোডের (in-memory code) এক্সিকিউশন ব্লক করে দেয় এবং ডিফল্টরূপে Vanadium ব্রাউজারে JavaScript Just-In-Time (JIT) কম্পাইলার ডিজেবল করে রাখে। এর ফলে জিরো-ডে

এক্সপ্লয়েটগুলোর (zero-day exploits) ব্যবহৃত মেমরি করাপশন ভালনারেবিলিটির (memory corruption vulnerabilities) সম্পূর্ণ ক্লাসটিই এলিমিনেট (eliminate) হয়ে যায়।

- **Aggressive Memory Wiping:** হার্ডেনড মেমরি অ্যালোকেটর (`hardened_malloc`) মেমরি ব্যবহার শেষ হওয়ার সাথে সাথে—ঠিক সেই মিলি-সেকেন্ডেই—মেমরি জিরো-আউট (zeroes out) করে দেয়। এটি নিশ্চিত করে যে কোনো অ্যাটাকার যদি সাময়িক কোনো ফুটহোল্ড (foothold) বা নিয়ন্ত্রণ পেয়েও যায়, তবুও সে যেন কোনো সেনসিটিভ ডেটা এক্সট্রাক্ট করতে না পারে।

দ্য ডিসিপ্লিন অফ দ্য অপারেটর (The Discipline of the Operator)

সফটওয়্যার ইনভিজিবল বা অদৃশ্য, লো-লেভেলের (low-level) থ্রেটগুলো হ্যান্ডেল করছে। কিন্তু হাই-লেভেল (high-level) অ্যাটাক সারফেস রিডাকশন সম্পূর্ণরূপে আপনার উপর নির্ভরশীল।

আমরা যখন ফোর্ট্রেস নির্মাণের দিকে অগ্রসর হচ্ছি, আপনাকে অবশ্যই এই আনকম্প্রোমাইজিং ডিসিপ্লিনটি অ্যাডপ্ট করতে হবে। আপনাকে আপনার ডিজিটাল জীবনকে বিভিন্ন আইসোলেটেড কম্পার্টমেন্টে (isolated compartments) বিভক্ত করতে বলা হবে। আপনাকে শেখানো হবে কীভাবে আপনার নেটওয়ার্ক ট্রাফিক রাউট করে আপনার ফিজিক্যাল অরিজিন বা অবস্থান হাইড করবেন। কর্মক্ষেত্র বা যোগাযোগের স্বার্থে আপনি যেসব হোস্টাইল অ্যাপ্লিকেশন ব্যবহার করতে বাধ্য, সেগুলোকে কীভাবে হ্যান্ডেল করতে হবে সে বিষয়েও আপনাকে ইনস্ট্রাকশন দেওয়া হবে।

আগামী চ্যাপ্টারগুলোতে আমরা যে প্রোফাইলগুলো বিল্ড করব এবং যে সেটিংগুলো কনফিগার করব, সেগুলোর প্রতিটিতেই একটি স্টক, কমার্শিয়াল ফোন ব্যবহারের চেয়ে কিছুটা বেশি এফোর্টের প্রয়োজন হবে। আপনার মুখের উপর সম্পূর্ণরূপে নির্ভর করার পরিবর্তে আপনাকে পাসওয়ার্ড এন্টার করতে হবে। বিভিন্ন অ্যাপ অ্যাক্সেস করতে আপনাকে প্রোফাইল সুইচ করতে হবে। আপনাকে ম্যানুয়ালি আপনার পারমিশনগুলো ম্যানেজ করতে হবে।

এই ফ্রিকশনকে (friction) হাসিমুখে মেনে নিন। এই ফ্রিকশনই হলো সিকিউরিটির ফিজিক্যাল সেনসেশন বা ভৌত অনুভূতি। যখন কোনো কাজে কয়েক সেকেন্ড অতিরিক্ত সময় লাগবে, তখন নিজেকে মনে করিয়ে দিন যে এই কয়েক সেকেন্ডের ব্যারিয়ার বা প্রতিরোধটুকুই একজন অ্যাটাকারকে নীরবে আপনার সারা জীবনের কাজগুলো এক্সট্রাক্ট করে নেওয়া থেকে আটকে রাখছে।

ফোর্ট্রেস বা দুর্গে আপনাকে স্বাগতম। চলুন, দেয়ালগুলো নির্মাণ শুরু করা যাক।

২.১ ব্রোকিং ছাড়া ডি-গুগলিং: Sandboxed Google Play আর্কিটেকচার

যখন কোনো ব্যক্তি তার ডিজিটাল প্রাইভেসি পুনরুদ্ধারের সিদ্ধান্ত নেন, তখন তার প্রথম প্রবৃত্তি হয় নিজের জীবন থেকে Google-কে সম্পূর্ণভাবে মুছে ফেলা। এই প্রক্রিয়াটি, যা সাধারণত "De-Googling" নামে পরিচিত, ঐতিহাসিকভাবে একটি মারাত্মক অপারেশনাল দ্বিধা বা ডাইলেমা তৈরি করত।

Android ইকোসিস্টেম Google Play Services-এর সাথে ওতপ্রোতভাবে জড়িত। Play Services নিছক কোনো অ্যাপ স্টোর নয়; এটি একটি বিশাল, অত্যন্ত প্রিভিলেজড ব্যাকগ্রাউন্ড ফ্রেমওয়ার্ক যা পুশ নোটিফিকেশন এবং লোকেশন ট্র্যাকিং থেকে শুরু করে ইন-অ্যাপ পারচেজ এবং সিকিউরিটি অ্যাটেস্টেশন পর্যন্ত সবকিছু পরিচালনা করে। আপনি যদি কোনো কাস্টম রম ফ্ল্যাশ করেন এবং Google Play Services ইনস্টল করতে অস্বীকৃতি জানান, তাহলে আপনি প্রাইভেসি তো পাবেন, কিন্তু ব্যবহারযোগ্যতার দিক থেকে আপনাকে চরম মূল্য দিতে হবে। আপনার ব্যাংকিং অ্যাপগুলো চালু হওয়ার সাথেই ক্র্যাশ করবে, আপনার সিকিউর মেসেঞ্জারগুলো ব্যাকগ্রাউন্ডে নোটিফিকেশন দিতে ব্যর্থ হবে, এবং রাইড-শেয়ারিং অ্যাপ্লিকেশনগুলো একেবারেই অকার্যকর হয়ে পড়বে।

বহু বছর ধরে, প্রাইভেসি কমিউনিটি এই বিশাল পরিমাণ কার্যকারিতা হারানোকে স্বাধীনতার জন্য একটি প্রয়োজনীয় ত্যাগ হিসেবে মেনে নিয়েছিল। আপনাকে দুটি বিষয়ের একটি বেছে নিতে বাধ্য করা হতো: একটি অকার্যকর ফোনের সাথে পরম প্রাইভেসি, অথবা এমন একটি কার্যকর ফোন যা আপনার প্রতিটি পদক্ষেপ সার্ভেইল বা নজরদারি করে।

GrapheneOS এই ভ্রান্ত বিভাজনকে প্রত্যাখ্যান করেছে।

এর **Sandboxed Google Play compatibility layer** নামক একটি ইঞ্জিনিয়ারিং মাস্টারস্ট্রোকের মাধ্যমে, GrapheneOS সম্পূর্ণভাবে "De-Googling" ডাইলেমার সমাধান করেছে। এটি আপনাকে Play Services প্রয়োজন হয় এমন কমার্শিয়াল Android অ্যাপ্লিকেশনগুলোর বিশাল ইকোসিস্টেম ব্যবহার করার সুযোগ দেয়, এবং একইসাথে অপারেটিং সিস্টেমের ওপর সার্ভেইল্যান্স চালানোর জন্য Google-এর সক্ষমতাকে সম্পূর্ণভাবে নির্মূল করে দেয়।

এই আর্কিটেকচারের অসাধারণত্ব বোঝার জন্য, আমাদেরকে অবশ্যই এর ঐতিহাসিক বিকল্পগুলোর সাথে এর পার্থক্য মূল্যায়ন করতে হবে।

সমস্যা: স্ট্যান্ডার্ড Android আর্কিটেকচার

একটি স্টক কমার্শিয়াল ডিভাইসে (যেমন একটি স্ট্যান্ডার্ড Google Pixel বা Samsung Galaxy), Google Play Services কোনো সাধারণ অ্যাপ্লিকেশন হিসেবে রান করে না। এটি অপারেটিং সিস্টেমের একেবারে গভীরতম স্তরে একটি অত্যন্ত প্রিভিলেজড সিস্টেম অ্যাপ হিসেবে হার্ডকোড করা থাকে।

এই প্রিভিলেজড ইন্টিগ্রেশনের কারণে, Google Play Services স্ট্যান্ডার্ড Android App Sandbox-কে বাইপাস করে। আপনার ডিভাইসের ওপর এর সীমাহীন এবং অপরিবর্তনীয় অ্যাক্সেস রয়েছে। এটি আপনার লোকেশন দেখতে পারে, আপনার হার্ডওয়্যার আইডেন্টিফায়ারগুলোতে অ্যাক্সেস করতে পারে, আপনার কন্ট্যাক্ট লিস্ট পড়তে পারে, এবং আপনার অনুমতি ছাড়াই ব্যাকগ্রাউন্ডে নীরবে অ্যাপ্লিকেশন ইনস্টল বা আনইনস্টল করতে পারে। এটি এমন একটি টেলিমেট্রি ইঞ্জিন যা প্রায় রুট-লেভেল অথরিটি নিয়ে কাজ করে। আপনি এটিকে আনইনস্টল করতে পারবেন না, এর পারমিশনগুলো সীমাবদ্ধ করতে পারবেন না, এবং Google-এর সার্ভারগুলোর সাথে এর যোগাযোগ করা বা ফোন হোম করা থেকে একে থামাতে পারবেন না।

ত্রুটিপূর্ণ ওয়ার্কঅ্যারাউন্ড: microG এবং Signature Spoofing

GrapheneOS তাদের কম্প্যাটিবিলিটি লেয়ার আর্কিটেক্ট করার আগে, কাস্টম রম কমিউনিটিতে (যেমন LineageOS) সবচেয়ে জনপ্রিয় ওয়ার্কঅ্যারাউন্ড ছিল **microG** নামক একটি প্রজেক্ট।

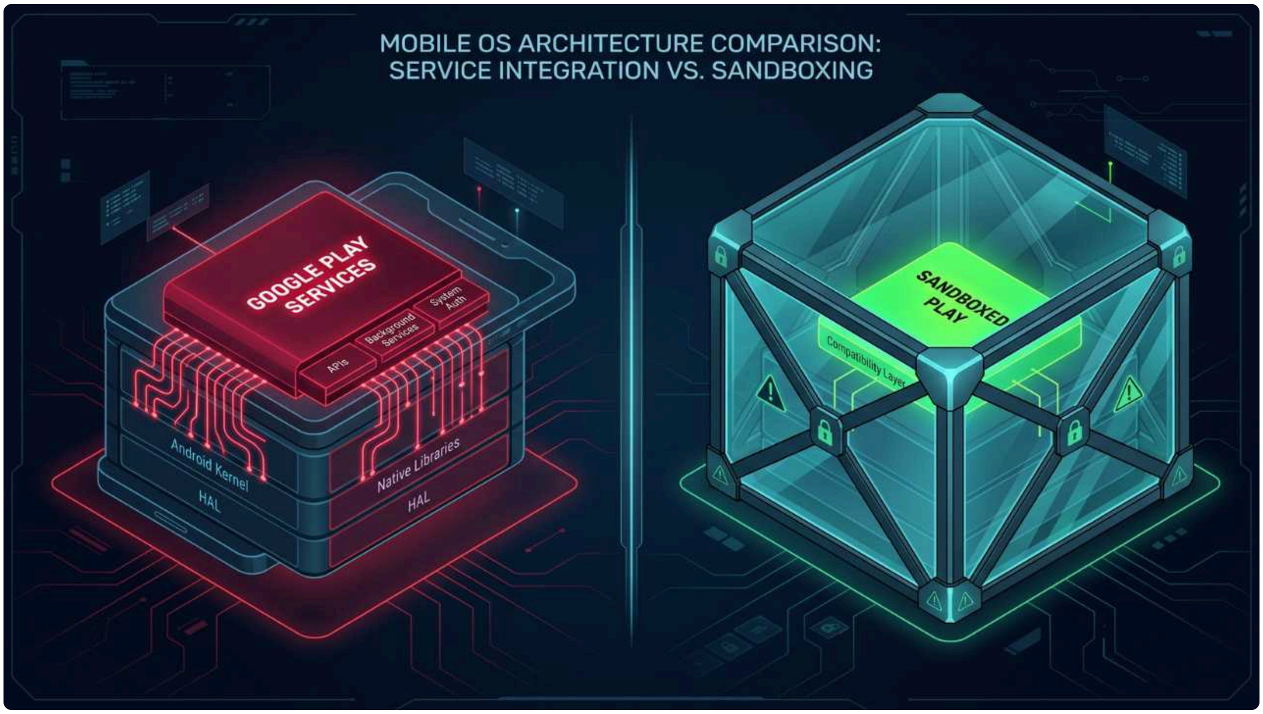
microG হলো Google Play Services-এর একটি ওপেন-সোর্স, রিভার্স-ইঞ্জিনিয়ারড ক্লোন। এটি Google-এর কাছে টেলিমেট্রি না পাঠিয়েই অ্যাপগুলোর প্রয়োজনীয় API-গুলোকে মিমিক করার চেষ্টা করে। যদিও এর ধারণাটি মহৎ, তবে এটি এক্সিকিউট করতে গেলে একটি মারাত্মক সিকিউরিটি কম্প্রোমাইজ করতে হয় যা **Signature Spoofing** নামে পরিচিত।

কমার্শিয়াল অ্যাপগুলো Play Services-এর ক্রিপ্টোগ্রাফিক সিগনেচার চেক করে নিশ্চিত হয় যে তারা আসল Google ফ্রেমওয়ার্কের সাথে যোগাযোগ করছে, কোনো ম্যালিসিয়াস ইমপোস্টারের সাথে নয়। যেহেতু microG Google-এর দ্বারা সাইন করা নয়, তাই microG-কে Google-এর অফিসিয়াল অ্যাপ হিসেবে দেখানোর জন্য কাস্টম ROM-কে ইচ্ছাকৃতভাবে Android-এর সিকিউরিটি মডেল পরিবর্তন করতে হয়, যাতে এটি Google-এর সিগনেচার "স্পুফ" বা নকল করতে পারে।

Signature Spoofing এনাবল করার মাধ্যমে, আপনি ইচ্ছাকৃতভাবে ক্রিপ্টোগ্রাফিক সত্যের ব্যাপারে অপারেটিং সিস্টেমকে অন্ধ করে দিচ্ছেন। যদি কোনো অ্যাডভার্সারি এমন একটি সিস্টেমে ম্যালিসিয়াস অ্যাপ্লিকেশন ইনস্টল করতে সক্ষম হয় যেখানে সিগনেচার স্পুফিং এনাবল করা আছে, তবে সেই ম্যালওয়্যারটি আপনার ব্যাংকিং অ্যাপ বা পাসওয়ার্ড ম্যানেজারের সিগনেচার স্পুফ করতে পারবে। Google-এর কাছ থেকে প্রাইভেসি পাওয়ার চেষ্টায়, microG-এর ব্যবহারকারীরা অনিচ্ছাকৃতভাবে তাদের সম্পূর্ণ ডিভাইসের ক্রিপ্টোগ্রাফিক ইন্টিগ্রিটিকে ধ্বংস করে ফেলে।

মাস্টারস্ট্রোক: GrapheneOS কম্প্যাটিবিলিটি লেয়ার

GrapheneOS এই উভয় অ্যাপ্রোচের সাথে মৌলিকভাবে দ্বিমত পোষণ করে। এটি Google-কে সিস্টেম-লেভেল প্রিভিলেজ দিতে অস্বীকৃতি জানায়, এবং সিগনেচার স্পুফিংয়ের মাধ্যমে অপারেটিং সিস্টেমকে কম্প্রোমাইজ করতেও অস্বীকার করে।



এর পরিবর্তে, GrapheneOS এমন একটি কম্প্যাটিবিলিটি লেয়ার তৈরি করেছে যা *অফিসিয়াল*, অপরিবর্তিত Google Play Services বাইনারিগুলোকে সম্পূর্ণ স্ট্যান্ডার্ড, আনপ্রিভিলেজড অ্যাপ হিসেবে রান করতে বাধ্য করে।

আপনি যখন GrapheneOS "Apps" রিপোজিটরি থেকে Google Play Services ইনস্টল করেন, তখন অপারেটিং সিস্টেম এটিকে সিস্টেম লেয়ারে ইন্টিগ্রেট করে না। এটি Google Play Services-কে ঠিক সেই একই রেস্ট্রিক্টিভ App Sandbox-এর ভেতরে ট্র্যাপ করে যা একটি ক্যালকুলেটর বা ওয়েদার অ্যাপের জন্য ব্যবহার করা হয়।

Google Play Services হঠাৎ নিজেকে একটি প্রতিকূল পরিবেশের মধ্যে আবিষ্কার করে। এটি ডিভাইসের হার্ডওয়্যার আইডেন্টিফায়ারগুলো অ্যাক্সেস করার চেষ্টা করে, এবং অপারেটিং সিস্টেম তাকে ব্লক করে দেয়। এটি ব্যাকগ্রাউন্ডে নীরবে একটি অ্যাপ ইনস্টল করার চেষ্টা করে, এবং অপারেটিং সিস্টেম তাকে ব্লক করে দেয়। এটি ধরে নেয় যে এর সম্পূর্ণ নিয়ন্ত্রণ রয়েছে, কিন্তু বাস্তবে এর কাছে কোনো ক্ষমতাই নেই।

সাধারণত, এই সিস্টেম-লেভেল প্রিভিলেজগুলো ডিনাই করা হলে Google Play Services সাথে সাথেই ক্র্যাশ করত, এবং এর ওপর নির্ভরশীল সমস্ত অ্যাপকেও সাথে নিয়ে ডাউন হয়ে যেত। এখানেই GrapheneOS কম্প্যাটিবিলিটি লেয়ারের আসল জাদু। এই লেয়ারটি ডায়নামিকভাবে এই প্রিভিলেজড রিকোয়েস্টগুলোকে ক্যাচ করে, ইন্টারসেপ্ট করে, এবং সেগুলোকে স্ট্যান্ডার্ড, আনপ্রিভিলেজড রিকোয়েস্টে ট্রান্সলেট করে দেয়। এটি মূলত Google Play Services-কে "শিক্ষা" দেয় যে কীভাবে ক্র্যাশ না করে একটি সাধারণ অ্যাপ হিসেবে টিকে থাকতে হয়।

এর ফলাফল অত্যন্ত গভীর। Google Play Services এই ভ্রান্ত ধারণার ওপর কাজ করে যে এটি এখনও একটি সিস্টেম অ্যাপ, সফলভাবে আপনার নির্ভরশীল অ্যাপগুলোতে API সরবরাহ করে, অথচ বাস্তবে এটি শূন্য প্রিভিলেজসহ একটি ক্রিপ্টোগ্রাফিক খাঁচায় আবদ্ধ থাকে। Google Play Services-এর যদি আপনার লোকেশন প্রয়োজন হয়, তবে তাকে একটি ইউজার-ফেসিং প্রম্পটের মাধ্যমে আপনার কাছে স্ট্যান্ডার্ড Location পারমিশন চাইতে হবে। যদি এটি কোনো অ্যাপ ইনস্টল করতে চায়, তবে তাকে আপনার স্পষ্ট সম্মতি বা কনসেন্ট চাইতে হবে।

ফাংশনাল প্যারিটি অর্জন

microG-এর মতো রিভার্স-ইঞ্জিনিয়ারড ক্লোনের ওপর নির্ভর করার পরিবর্তে একটি স্যান্ডবক্সের মধ্যে অফিসিয়াল Google বাইনারিগুলো রান করার মাধ্যমে, GrapheneOS কমার্শিয়াল Android-এর সাথে প্রায় ১০০% ফাংশনাল প্যারিটি অর্জন করে। স্পাইং ছাড়াই সম্পূর্ণ ইকোসিস্টেমটি নিখুঁতভাবে কাজ করে।

- **FCM (Firebase Cloud Messaging):** পুশ নোটিফিকেশন হলো মোবাইল কমিউনিকেশনের প্রাণশক্তি। Signal-এর মতো সিকিউর মেসেঞ্জারগুলো ব্যাটারি ড্রেন না করে ডিভাইসটিকে ওয়েক আপ করতে এবং এনক্রিপ্টেড মেসেজগুলো দক্ষতার সাথে ডেলিভার করতে FCM-এর ওপর নির্ভর করে। Sandboxed Google Play নেটিভভাবে FCM হ্যান্ডেল করে। আপনি মেসেজ কন্টেন্ট বা সিস্টেম-লেভেল মেটাডেটায় Google-কে অ্যাক্সেস না দিয়েই সিকিউর মেসেঞ্জার, ব্যাংকিং অ্যালার্ট এবং ইমেইলের জন্য তাৎক্ষণিক পুশ নোটিফিকেশন পান।
- **Dynamite Modules:** অনেক আধুনিক অ্যাপ কাজ করার জন্য Play Services থেকে ডায়নামিকভাবে ডাউনলোড করা কোড মডিউলগুলোর (Dynamite modules) ওপর নির্ভর করে—যেমন টিকেটিং অ্যাপগুলোতে ব্যবহৃত বারকোড স্ক্যানিং API অথবা রাইড-শেয়ারিং অ্যাপের ভেতরে থাকা Google Maps রেন্ডারিং লাইব্রেরিগুলো। Sandboxed Google Play এই মডিউলগুলোকে এর আনপ্রিভিলেজড স্যান্ডবক্সের ভেতরে নিখুঁতভাবে ডাউনলোড এবং এক্সিকিউট করে।
- **Play Asset ও Feature Delivery:** যেসব ভারী অ্যাপ্লিকেশন এবং কমার্শিয়াল গেমস Play Asset Delivery-এর (ব্যাকগ্রাউন্ডে বড় গ্রাফিক্স বা লেভেল ডাউনলোড করা) ওপর নির্ভর করে, সেগুলো স্টক ডিভাইসে ঠিক যেভাবে চলত সেভাবেই কাজ করে।

- **In-App Purchases ও Licensing:** Sandboxed Google Play সম্পূর্ণরূপে Google Play Store বিলিং API সাপোর্ট করে। আপনি আপনার অপারেটিং সিস্টেমের ইন্টিগ্রিটির সাথে আপস না করেই নিরাপদে অ্যাপ্লিকেশন কিনতে, সাবস্ক্রিপশন ম্যানেজ করতে এবং পেইড সফটওয়্যারের জন্য ক্রিপ্টোগ্রাফিক লাইসেন্স চেকগুলো পাস করতে পারবেন।
- **Geolocation Rerouting:** ডিফল্টরূপে, Sandboxed Google Play Google-এর ইনভেসিভ নেটওয়ার্ক লোকেশন সার্ভিসে পাঠানো রিকোয়েস্টগুলোকে ইন্টারসেপ্ট করে এবং সেগুলোকে নিরাপদে GrapheneOS-এর প্রাইভেসি-রেসপেক্টিং, লোকালি ক্যাশ করা লোকেশন সার্ভিসে রিরাউট করে।

আর্কিটেকচারাল কম্পারিজন

আর্কিটেকচারাল কম্পারিজন: স্ট্যান্ডার্ড Android বনাম microG বনাম GrapheneOS

Play Services প্রিভিলেজ লেভেল

- স্ট্যান্ডার্ড কমার্শিয়াল Android: হাইলি প্রিভিলেজড সিস্টেম অ্যাপ
- কাস্টম রম (microG): সিস্টেম-লেভেল মক সার্ভিস
- GrapheneOS (Sandboxed Google Play): স্ট্যান্ডার্ড, আনপ্রিভিলেজড অ্যাপ

Signature Spoofing প্রয়োজন কি?

- স্ট্যান্ডার্ড কমার্শিয়াল Android: না (অফিসিয়াল কি ব্যবহার করে)
- কাস্টম রম (microG): **হ্যাঁ (সিকিউরিটি মডেল ভেঙে দেয়)**
- GrapheneOS (Sandboxed Google Play): না (একটি স্যান্ডবক্সের ভেতর অফিসিয়াল কি ব্যবহার করে)

App Sandbox বাইপাস করে কি?

- স্ট্যান্ডার্ড কমার্শিয়াল Android: হ্যাঁ
- কাস্টম রম (microG): হ্যাঁ
- GrapheneOS (Sandboxed Google Play): **না (কঠোরভাবে আইসোলেটেড)**

ব্যাকগ্রাউন্ড সাইলেন্ট ইনস্টল?

- স্ট্যান্ডার্ড কমার্শিয়াল Android: হ্যাঁ (কোনো ইউজার কনসেন্ট প্রয়োজন নেই)
- কাস্টম রম (microG): ইমপ্লিমেন্টেশনের ওপর ভিত্তি করে ভিন্ন হয়

- GrapheneOS (Sandboxed Google Play): **না (স্পষ্ট ইউজার কনসেন্ট প্রয়োজন)**

হার্ডওয়্যার আইডেন্টিফায়ার অ্যাক্সেস?

- স্ট্যান্ডার্ড কমার্শিয়াল Android: আনরেস্ট্রিক্টেড অ্যাক্সেস (IMEI, Serial)
- কাস্টম রম (microG): মকড/স্পুফড
- GrapheneOS (Sandboxed Google Play): **অপারেটিং সিস্টেম দ্বারা সম্পূর্ণভাবে ব্লকড**

অ্যাপ কম্প্যাটিবিলিটি

- স্ট্যান্ডার্ড কমার্শিয়াল Android: ১০০%
- কাস্টম রম (microG): প্রায়ই কাজ করে না (Banking, Maps)
- GrapheneOS (Sandboxed Google Play): প্রায় ১০০%

Sandboxed Google Play আর্কিটেকচারটি হলো De-Googling ডাইলেমার চূড়ান্ত সমাধান। পরম প্রাইভেসি এবং একটি ফাংশনাল ডিভাইসের মধ্যে আপনাকে আর কোনোটি বেছে নিতে হবে উল্লেখযোগ্য। একটি সুরক্ষিত খাঁচা তৈরি করার মাধ্যমে, GrapheneOS আপনাকে দুর্গের চাবি হস্তান্তর না করেই করপোরেট ইকোসিস্টেমের সুবিধা উপভোগ করার সুযোগ দেয়।

> *[!TIP] **ভেরিফাইড অ্যাপ ডিরেক্টরি (Verified App Directory)***
> *যদিও Sandboxed Google Play আপনাকে কমার্শিয়াল অ্যাপগুলো নিরাপদে ব্যবহার করার সুযোগ দেয়, তবুও আমরা সম্ভব হলে সেগুলোর পরিবর্তে FOSS (ফ্রি এবং ওপেন-সোর্স) বিকল্প ব্যবহার করার তীব্র পরামর্শ দিই। আপনার দৈনন্দিন অ্যাপগুলোর ১০০% ভেরিফাইড FOSS বিকল্প খুঁজে পেতে এবং যেসব হাই-রিস্ক টেলিমেট্রি অ্যাপ আপনার স্যান্ডবক্সেও ইনস্টল করা উচিত নয় সেগুলোর তালিকা দেখতে আমাদের অফিসিয়াল GrapheneOS অ্যাপ্লিকেশন ডিরেক্টরি ভিজিট করুন।*

২.২ ফ্যাকিং আউট Google: Location এবং Network Routing

ইন্টেলিজেন্স কমিউনিটিতে, কোনো টার্গেটের শারীরিক গতিবিধি বা ফিজিক্যাল মুভমেন্ট ট্র্যাক করা হচ্ছে সবচেয়ে মূল্যবান কালেকশন মেট্রিক। এটি সংশ্লিষ্টতা, রুটিন এবং দুর্বলতা বা ভালনারেবিলিটিগুলোকে প্রকাশ করে। কমার্শিয়াল টেকনোলজি সেক্টরেও ঠিক একই জিনিস ঘটে, তবে এখানে উদ্দেশ্যটা ট্যাক্টিক্যালের বদলে মূলত ফিন্যান্সিয়াল।

Google-এর ভৌত জগৎ বা ফিজিক্যাল ওয়ার্ল্ড ম্যাপ করার প্রাথমিক মেকানিজম হলো Fused Location Provider (FLP)। একটি স্ট্যান্ডার্ড Android এনভায়রনমেন্টে, অ্যাপ্লিকেশনগুলো সরাসরি GPS চিপের কাছে কো-অর্ডিনেট বা স্থানাঙ্ক চায় এমন ঘটনা বিরল। এর বদলে তারা Google Play Services-এর কাছে তা জানতে চায়। Play Services এখানে একজন মিডলম্যান বা মধ্যস্বত্বভোগী হিসেবে কাজ করে, যা GPS স্যাটেলাইট, কাছাকাছি থাকা Wi-Fi MAC অ্যাড্রেস, Bluetooth বীকন এবং সেলুলার টাওয়ারগুলো থেকে ডেটা পুল করে। এটি এই ডেটাগুলো অ্যাগ্রিগেট বা একত্রিত করে, আপনার সুনির্দিষ্ট লোকেশন ক্যালকুলেট করে, রিকোয়েস্ট করা অ্যাপের কাছে তা হস্তান্তর করে, এবং একইসাথে Google-এর রিমোট সার্ভারগুলোতে আপনার লোকেশন হিস্ট্রি লগ করে রাখে।

আপনি যদি ২.১ সেকশনের নির্দেশনাগুলো অনুসরণ করে থাকেন, তবে আপনি Google Play Services-কে একটি স্যান্ডবক্সের ভেতর আটকে ফেলেছেন। কিন্তু একটি স্যান্ডবক্সড Play Services এখনও লোকেশন মিডলম্যান হিসেবে কাজ করার প্রত্যাশা করে। যদি কোনো রাইড-শেয়ারিং অ্যাপ স্যান্ডবক্সড Play Services-এর কাছে লোকেশন চায়, এবং Play Services সেই রিকোয়েস্ট পূরণ করতে না পারে, তবে রাইড-শেয়ারিং অ্যাপটি ক্র্যাশ করবে অথবা ম্যাপ রেন্ডার করতে ব্যর্থ হবে।

GrapheneOS এই সমস্যার সমাধান করেছে একটি দুর্দান্ত ক্রিপ্টোগ্রাফিক স্লাইট-অফ-হ্যান্ড বা কারসাজির মাধ্যমে: **Location Redirection.**

Location Redirection-এর মেকানিক্স

যখন আপনি GrapheneOS App রিপোজিটরি ব্যবহার করে Google Play Services ইনস্টল করেন, তখন অপারেটিং সিস্টেম স্বয়ংক্রিয়ভাবে একটি নির্দিষ্ট কম্প্যাটিবিলিটি ফিচার এনাবল করে, যার নাম হলো "Reroute location requests to the OS."

যখন কোনো অ্যাপ্লিকেশন (যেমন Uber বা ওয়েদার অ্যাপ) আপনার লোকেশন রিকোয়েস্ট করে, তখন সেই রিকোয়েস্টটি স্যান্ডবক্সড Google Play Services-এর কাছে পাঠানো হয়। তবে, Play Services এর প্রোপ্রাইটারি, ডেটা-হারভেস্টিং Fused Location Provider অ্যালগরিদম এক্সিকিউট করার আগেই, GrapheneOS কম্প্যাটিবিলিটি লেয়ার নীরবে

রিকোয়েস্টটি ইন্টারসেপ্ট করে।

GrapheneOS তৎক্ষণাৎ রিকোয়েস্টটিকে Google-এর কোড থেকে দূরে সরিয়ে দেয় এবং সরাসরি তার নিজস্ব, হাইলি সিকিউর, প্রাইভেট OS-লেভেল লোকেশন প্রোভাইডারের কাছে রিরাউট করে। OS আপনার ফিজিক্যাল কো-অর্ডিনেটগুলো লোকালি ক্যালকুলেট করে—কোম্পানির কোনো সার্ভারে আপনার অবস্থান রিপোর্ট না করে বা আপনার হিস্ট্রি লগ না করেই। এরপর এটি এই কো-অর্ডিনেটগুলো Google Play Services-কে ফেরত দেয়, যা পরবর্তীতে রিকোয়েস্ট করা অ্যাপটির কাছে তা হস্তান্তর করে।

Google Play Services এই ভ্রান্ত ধারণার ওপর কাজ করে যে এটি সফলভাবে লোকেশন রিকোয়েস্ট প্রসেস করেছে, অথচ বাস্তবে একে সম্পূর্ণভাবে বাইপাস করা হয়েছিল। নির্ভরশীল অ্যাপটি কাজ করার জন্য প্রয়োজনীয় কো-অর্ডিনেটগুলো পেয়ে যায়, এবং আপনার ফিজিক্যাল মুভমেন্ট হিস্ট্রি একচেটিয়াভাবে আপনার লোকাল হার্ডওয়্যারেই থেকে যায়। এর মাধ্যমে আপনি কার্যকরভাবে Google-এর সার্ভেইল্যান্স আর্কিটেকচারকে ফেইক আউট করেছেন বা ধোঁকা দিয়েছেন।

ব্যতিক্রম: Google Maps Live Location Sharing

অপারেশনাল সিকিউরিটি (OpSec) হলো একটি স্পেকট্রাম, এবং মাঝে মাঝে অপারেটরদের প্রয়োজনীয় ট্যাক্টিক্যাল উদ্দেশ্য অর্জনের জন্য ক্যালকুলেটেড রিস্ক গ্রহণ করতে হয়।

নতুন GrapheneOS ব্যবহারকারীদের জন্য সবচেয়ে সাধারণ একটি ঘর্ষণ বা ফ্রিকশন পয়েন্ট হলো লাইভ লোকেশন শেয়ারিং। পরিবার, এক্সিকিউটিভ প্রটেকশন টিম, বা দম্পতিরা একে অপরের সাথে লাইভ, রিয়েল-টাইম লোকেশন শেয়ার করার জন্য প্রায়ই Google Maps-এর ওপর নির্ভর করে।

আপনি যদি ডিফল্ট GrapheneOS সেটিংসে Google Maps-এ "Location Sharing" ব্যবহার করার চেষ্টা করেন, তবে এটি ব্যর্থ হবে। কেন? কারণ Google-এর রিমোট ব্যাকএন্ড সার্ভারগুলো সরাসরি অফিসিয়াল Google Play Services Fused Location Provider থেকে লোকেশন ডেটার একটি একটানা, ক্রিপ্টোগ্রাফিক্যালি সাইন করা স্ট্রিম বা প্রবাহ প্রত্যাশা করে। যেহেতু GrapheneOS সেই রিকোয়েস্টগুলোকে ইন্টারসেপ্ট করে এবং লোকালি রিরাউট করে, তাই Google-এর সার্ভারগুলো সেই ডেটা স্ট্রিম প্রত্যাখ্যান করে, এবং আপনার পরিবারের ডিভাইসগুলোতে আপনার লাইভ লোকেশন আপডেট হবে না।

আপনার থ্রেট মডেল যদি Google Maps Live Location Sharing ব্যবহারের দাবি করে, তবে আপনাকে অবশ্যই ম্যানুয়ালি আপনার ডিফেন্স বা প্রতিরোধ ব্যবস্থা কমিয়ে আনতে হবে এবং আপনাকে ট্র্যাক করার জন্য Google-কে সুস্পষ্ট পারমিশন দিতে হবে।

The Configuration Protocol (কনফিগারেশন প্রোটোকল):

1. নেভিগেট করুন **Settings** > **Apps** > **Sandboxed Google Play**-তে।
2. **Reroute location requests to the OS** লেবেলযুক্ত টগলটিতে ট্যাপ করুন এবং এটি **OFF** করে দিন।
3. নেভিগেট করুন **Settings** > **Apps** > **Google Play services** > **Permissions**-এ।
4. **Location**-এ ট্যাপ করুন এবং পারমিশন পরিবর্তন করে **Allow all the time** করুন।

> ⚠ ***Warning:*** *এই ধাপগুলো সম্পাদন করার মাধ্যমে, আপনি ব্যাকগ্রাউন্ডে আপনার ডিভাইসের লোকেশন মনিটর করার এবং সেই টেলিমেট্রি তাদের সার্ভারগুলোতে ট্রান্সমিট করার জন্য Google Play Services-কে সক্রিয়ভাবে অথোরাইজেশন দিচ্ছেন। আপনি কেবলমাত্র কনভেনিয়েন্স বা সুবিধার খাতিরে GrapheneOS লোকেশন শিল্ড বা ঢালের পেছন থেকে বেরিয়ে আসছেন। নিশ্চিত করুন যে আপনি যার সাথে আপনার লোকেশন শেয়ার করছেন তিনি আপনার ফিজিক্যাল OpSec কম্প্রোমাইজ করার যোগ্য কি না।*

"Nearby Devices" এবং Bluetooth Beacon থ্রেট

শহরের এক প্রান্ত থেকে অন্য প্রান্তে টার্গেট ট্র্যাক করার ক্ষেত্রে GPS দারুণ কার্যকর, কিন্তু ইনডোর বা বদ্ধ ঘরের ভেতরে এটি ব্যর্থ হয়। শপিং মল, এয়ারপোর্ট এবং করপোরেট অফিসের ভেতরের ম্যাপ তৈরি করার জন্য, সার্ভেইল্যান্স ইন্ডাস্ট্রি সম্পূর্ণ ভিন্ন একটি টেকনোলজি ব্যবহার করে: Bluetooth Beacons।

হাজার হাজার রিটেইল স্টোর এবং করপোরেট বিল্ডিং তাদের আর্কিটেকচারের সর্বত্র লুকানো, লো-এনার্জি Bluetooth বীকন ডিপ্লয় করে। একটি স্ট্যান্ডার্ড কমার্শিয়াল OS-এ, আপনার ফোন ব্যাকগ্রাউন্ডে সারাক্ষণ এই বীকনগুলো স্ক্যান করতে থাকে। যখন আপনি কোনো মলের জুতার ডিপার্টমেন্টে একটি বীকনের পাশ দিয়ে হাঁটেন, তখন আপনার ফোন সেই ইন্টারঅ্যাকশনটি লগ করে, এবং আপনি জুতা দেখতে ঠিক ১৪ মিনিট ব্যয় করেছেন—তা প্রতিফলিত করার জন্য আপনার ডিজিটাল প্রোফাইল আপডেট করা হয়।

এই কাজটি সহজ করার জন্য, Android **Nearby Devices** নামক একটি পারমিশনের ওপর নির্ভর করে।

ঐতিহাসিকভাবে, "Nearby Devices" পারমিশনটি "Location" পারমিশনের সাথেই যুক্ত ছিল, যা এটি ম্যানেজ করা অবিশ্বাস্যরকম কঠিন করে তুলেছিল। GrapheneOS সঠিকভাবে এটিকে একটি সম্পূর্ণ আইসোলেটেড, হাইলি সেনসিটিভ পারমিশন হিসেবে এনফোর্স বা কার্যকর করে।

অনেক অ্যাপ লঞ্চ করার মুহূর্তেই অত্যন্ত আগ্রাসীভাবে "Nearby Devices" পারমিশন চাইবে। ফাস্ট-ফুড অ্যাপ, রিটেইল অ্যাপ এবং সোশ্যাল মিডিয়া অ্যাপগুলো দাবি করবে যে "আপনার অভিজ্ঞতা উন্নত করতে" তাদের এটি প্রয়োজন। এটি একটি মিথ্যা কথা। তারা বীকন স্ক্যানিংয়ের মাধ্যমে আপনার ইনডোর ফিজিক্যাল মুভমেন্ট ম্যাপ করার জন্য আপনার Bluetooth রেডিওকে ওয়েপোনাইজ বা অস্ত্রে পরিণত করার চেষ্টা করে।

The Rule of Engagement (এনগেজমেন্টের নিয়ম): আপনাকে অবশ্যই একটি সুনির্দিষ্ট ব্যতিক্রম ছাড়া, অন্য সকল কিছুর ক্ষেত্রে "Nearby Devices" পারমিশন ডিনাই করতে হবে।

একমাত্র যে সময়ে আপনার "Nearby Devices" পারমিশন দেয়া উচিত তা হলো যখন আপনি সক্রিয়ভাবে কোনো অ্যাপ্লিকেশনের সাথে একটি ফিজিক্যাল হার্ডওয়্যার পেয়ার বা যুক্ত করার চেষ্টা করছেন। উদাহরণস্বরূপ, আপনি যদি আপনার Wi-Fi নেটওয়ার্কে একটি স্মার্ট স্পিকার সেট আপ করেন, অথবা কোনো মেডিক্যাল ডিভাইসকে (যেমন কন্টিনিউয়াস গ্লুকোজ মনিটর) এর কম্প্যানিয়ন অ্যাপের সাথে পেয়ার করেন, তখন আপনাকে সাময়িকভাবে পারমিশনটি দিতে হবে।

একবার পেয়ারিং প্রক্রিয়া সম্পন্ন হয়ে গেলে, অ্যাপের পারমিশন সেটিংসে গিয়ে "Nearby Devices" পারমিশন রিভোক বা বাতিল করে দিন।

সিগন্যাল কন্ট্রোল করা

আপনার লোকেশন ম্যাপের ওপর কেবল একটি বিন্দু নয়; এটি আপনার জীবনের বাস্তব প্রতিফলন। Sandboxed Google Play-এর লোকেশন রিডিরেকশন কাজে লাগিয়ে, আপনি কোনো ডিজিটাল ভেপার ট্রেইল বা ধোঁয়ার দাগ না রেখেই আধুনিক বিশ্বে নেভিগেট করতে পারবেন। "Nearby Devices" পারমিশন কঠোরভাবে ম্যানেজ করার মাধ্যমে, আপনি ইনডোর সার্ভেইল্যান্স নেটওয়ার্কগুলোকে অন্ধ করে দেবেন, যেগুলো প্যাসিভ ডেটা হারভেস্টিংয়ের ওপর নির্ভর করে।

আপনি রেডিওগুলো কন্ট্রোল করেন। আপনি ডেটা ফ্লো ডিক্টেট করেন। আর দুর্গ থাকে দুর্ভেদ্য।

2.3 Storage Scopes: ফেইক স্টোরেজ দিয়ে অ্যাপগুলোকে ধোঁকা দেওয়া

আপনার ডিভাইসের সবচেয়ে মূল্যবান সম্পদ হলো আপনার আনস্ট্রাকচার্ড ডেটা (unstructured data)। হাইলি স্ট্রাকচার্ড ইন্টেলিজেন্স (highly structured intelligence)—যেমন আপনার সুনির্দিষ্ট GPS কোঅর্ডিনেটস (coordinates) বা কন্টাক্ট লিস্টের (contact list) মতো নয়—আপনার ফাইল সিস্টেম হলো আপনার সম্পূর্ণ ডিজিটাল অস্তিত্বের এক বিশৃঙ্খল ভাণ্ডার। এতে ডাউনলোড করা ব্যাংক স্টেটমেন্ট, এনক্রিপ্টেড ব্যাকআপ (encrypted backups), ব্যক্তিগত ছবি, ক্যাশড (cached) ওয়েব পেজ এবং PDF ট্যাক্স রিটার্ন অন্তর্ভুক্ত থাকতে পারে।

একটি হোস্টাইল অ্যাপ্লিকেশন (hostile application) বা আক্রমণাত্মক ডেটা ব্রোকারের (data broker) জন্য, আপনার ইন্টারনাল স্টোরেজে (internal storage) অ্যাক্সেস পাওয়াটা হলো তাদের চূড়ান্ত পুরস্কার।

দুর্ভাগ্যবশত, স্ট্যান্ডার্ড কমার্শিয়াল অপারেটিং সিস্টেমগুলো ঐতিহাসিকভাবেই এই ডেটা সুরক্ষিত রাখতে ব্যর্থ হয়েছে। GrapheneOS-এর ডিফেন্স সিস্টেমের (defenses) বুদ্ধিমত্তা বুঝতে হলে, আমাদের প্রথমে ট্র্যাডিশনাল Android পারমিশন মডেলের (permission model) মারাত্মক ত্রুটিটি বিশ্লেষণ করতে হবে।

স্টোরেজ পারমিশনের (Storage Permission) চাঁদাবাজি

একটি স্ট্যান্ডার্ড Android ডিভাইসে, ফাইল সিস্টেমের সাথে ইন্টার‍্যাক্ট (interact) করতে ইচ্ছুক কোনো অ্যাপ্লিকেশনকে অবশ্যই "Files and media" (বা ঐতিহাসিকভাবে, "Read/Write External Storage") পারমিশনের জন্য রিকোয়েস্ট করতে হয়।

এই পারমিশনটি বিপজ্জনকভাবে বিস্তৃত। যখন কোনো অ্যাপ্লিকেশন স্টোরেজ অ্যাক্সেসের (storage access) রিকোয়েস্ট করে, তখন মূলত এটি আপনার পুরো ইন্টারনাল হার্ড ড্রাইভের (internal hard drive) মাস্টার কি (master key) দাবি করে। আপনি যদি একটি সিঙ্গেল প্রোফাইল পিকচার (profile picture) আপলোড করার জন্য কোনো সোশ্যাল মিডিয়া অ্যাপকে স্টোরেজ পারমিশন দেন, তবে আপনি একই সাথে সেই অ্যাপ্লিকেশনটিকে আপনার সম্পূর্ণ ডিভাইসের প্রতিটি ছবি, ডকুমেন্ট এবং ডাউনলোড ফোল্ডার স্ক্যান (scan), রিড (read) এবং ইনডেক্স (index) করার অথোরাইজেশন (authorization) দিয়ে দিচ্ছেন।

আপনি যদি সিকিউর (secure) থাকতে চান এবং পারমিশন ডিনাই (deny) করেন, তবে অ্যাপ্লিকেশনটি সাধারণত আপনার সাথে এক ধরণের চাঁদাবাজি করবে। এটি একটি ফুল-স্ক্রিন (full-screen) ওয়ার্নিং প্রদর্শন করে দাবি করবে, "স্টোরেজ পারমিশন ছাড়া এই অ্যাপ কাজ করতে পারবে না," এবং যতক্ষণ না আপনি চাবি হস্তান্তর করছেন, এটি ফোর্সফুলি (forcefully) ক্র্যাশ (crash) করবে বা লোড হতে অস্বীকৃতি জানাবে।

আপনি একটি অজেয় পরিস্থিতিতে আটকা পড়েছেন: হয় আপনার সম্পূর্ণ ডিজিটাল জীবন সমর্পণ করুন, অথবা অ্যাপ্লিকেশনটি পুরোপুরি বর্জন করুন।

GrapheneOS-এর মাস্টারস্ট্রোক: Storage Scopes

GrapheneOS বুঝতে পেরেছে যে, ম্যালিশিয়াস (malicious) এবং পোরলি কোডেড (poorly coded) অ্যাপগুলো যদি তাদের কাঙ্ক্ষিত পারমিশন না পায়, তবে সেগুলো আক্রমণাত্মকভাবে ক্র্যাশ করে। সুতরাং, GrapheneOS সহজভাবে তাদের সাথে মিথ্যা বলার সিদ্ধান্ত নিয়েছে।

এটি **Storage Scopes** নামক একটি প্রোপাইটারি (proprietary), বৈপ্লবিক ফিচারের মাধ্যমে অর্জন করা হয়েছে।

Storage Scopes হলো একটি হাইলি অ্যাডভান্সড কম্প্যাটিবিলিটি লেয়ার (compatibility layer), যা একটি অ্যাপ্লিকেশনের স্টোরেজ রিকোয়েস্টগুলোকে ইন্টারসেপ্ট (intercept) করে। আপনি যখন কোনো নির্দিষ্ট অ্যাপের জন্য Storage Scopes এনাবল (enable) করেন, তখন GrapheneOS অ্যাপ্লিকেশনটিকে একটি গাণিতিক মিথ্যা বলে: এটি অ্যাপটিকে ইনফর্ম করে যে, ইউজার পুরো ফাইল সিস্টেমে ফুল এবং আনরেস্ট্রিক্টেড অ্যাক্সেস (unrestricted access) দিয়েছে। অ্যাপটি এতে সন্তুষ্ট হয়। এটি অভিযোগ করা বন্ধ করে, নিজস্ব সিকিউরিটি চেক (security checks) বাইপাস (bypass) করে এবং স্বাভাবিকভাবে অপারেট (operate) করতে শুরু করে।

যাইহোক, পর্দার আড়ালে, GrapheneOS অ্যাপ্লিকেশনটিকে একটি কঠোরভাবে নিয়ন্ত্রিত, অদৃশ্য স্যান্ডবক্সের (sandbox) ভেতরে আটকে ফেলে। যখন অ্যাপটি ডিভাইসের স্টোরেজ স্ক্যান করার চেষ্টা করে, তখন এটি আপনার ব্যাংক স্টেটমেন্ট বা ব্যক্তিগত ছবি দেখতে পায় না। এটি কেবল একটি সম্পূর্ণ শূন্যতা (void) দেখতে পায়।

Storage Scopes-এর চমৎকার দিক হলো এটি আপনাকে সেই শূন্যতার মধ্যে ডায়নামিক্যালি (dynamically) অতি-ক্ষুদ্র, হাইলি স্পেসিফিক হোলস (highly specific holes) তৈরি করার অনুমতি দেয়। আপনি একটি অ্যাপ্লিকেশনকে রেস্ট্রিক্ট (restrict) করতে পারেন যাতে এটি শুধুমাত্র একটি সিঙ্গেল, স্পেসিফিক ফোল্ডার (specific folder) বা এমনকি একটি সিঙ্গেল, স্পেসিফিক ফাইল (specific file) দেখতে পায়।

একটি রিয়েল-ওয়ার্ল্ড সিনারিও: দ্য WhatsApp কোয়ারেন্টাইন (The WhatsApp Quarantine)

Storage Scopes-এর ট্যাকটিকাল পাওয়ার (tactical power) বোঝানোর জন্য WhatsApp-এর ডিপ্লয়মেন্ট (deployment) বিবেচনা করা যাক।

WhatsApp-এর মালিকানা মেটার (Meta) হাতে। একটি অপারেশনাল সিকিউরিটি (operational security) দৃষ্টিকোণ থেকে, এটি একটি হোস্টাইল (hostile) অ্যাপ্লিকেশন, যা নিরলসভাবে মেটাডেটা (metadata) হারভেস্ট (harvest) করে। যাইহোক, নেটওয়ার্ক ইফেক্টের (network effects) কারণে, অনেক অপারেটরকে তাদের অপেক্ষাকৃত কম সিকিউরিটি-সচেতন সহকর্মী বা পরিবারের সদস্যদের সাথে যোগাযোগ করতে বাধ্য হয়ে এটি ইনস্টল রাখতে হয়।

ধরে নিন, আপনি WhatsApp-এ কোনো ফ্যামিলি গ্রুপ চ্যাটে (group chat) একটি মজার ছবি বা "মিম" (meme) পাঠাতে চান।

আপনি যদি WhatsApp-কে স্ট্যান্ডার্ড Android স্টোরেজ পারমিশন দেন, তবে মেটার অ্যাপ্লিকেশনটি এখন আপনার মেইন ক্যামেরা রোল (camera roll), ডাউনলোড ফোল্ডার এবং আপনার ডিভাইসের সম্পূর্ণ হিস্ট্রির প্রতিটি ছবি ও ভিডিও স্ক্যান করার জন্য পারমানেন্ট ও আনরেস্ট্রিক্টেড অ্যাক্সেস (permanent, unrestricted access) পেয়ে গেল।

GrapheneOS-এর মাধ্যমে আমরা WhatsApp কোয়ারেন্টাইন এক্সিকিউট (execute) করতে পারি:

১. আপনার ডিভাইসে `Memes` নামে একটি স্পেসিফিক ফোল্ডার তৈরি করুন। ২. মজার ছবিটি সেই `Memes` ফোল্ডারের ভেতরে রাখুন। ৩. WhatsApp-এর জন্য Storage Scopes এনাবল (enable) করুন। ৪. `Memes` ফোল্ডারটিকে *একমাত্র* পারমিটেড স্কোপ (permitted scope) হিসেবে অ্যাড (add) করুন।

যখন WhatsApp ওপেন হয়, এটি মিডিয়া খোঁজার জন্য আপনার ফোনটি স্ক্যান করার চেষ্টা করে। GrapheneOS তখন WhatsApp-কে জানায় যে এটির কাছে ফুল অ্যাক্সেস (full access) রয়েছে, কিন্তু WhatsApp যখন চারপাশে তাকায়, তখন সে এই গোটা ইউনিভার্সে কেবল `Memes` ফোল্ডারটিকেই দেখতে পায়। এটি আপনার ক্যামেরা রোল বা ডাউনলোড ফোল্ডার দেখতে পারে না। আপনার পুরো ফাইল সিস্টেমটি ক্রিপ্টোগ্রাফিক্যালি ক্লোকড (cryptographically cloaked) থাকে, এবং WhatsApp পুরোপুরিভাবে আনঅ্যাওয়ার (unaware) থাকে যে তাকে ম্যানিপুলেট (manipulate) করা হচ্ছে। এটি পারফেক্টলি ফাংশন করে, মিমটি পাঠায়, এবং আপনার OpSec ইমপেনিট্রেবল (impenetrable) থেকে যায়।

কীভাবে Storage Scopes কনফিগার করবেন

ফাইল অ্যাক্সেসের রিকোয়েস্ট করা প্রতিটি অ্যাপ্লিকেশন ইনস্টল করার ক্ষেত্রে Storage Scopes ডিপ্লয় করাটা আপনার স্ট্যান্ডার্ড অপারেটিং প্রসিডিউর (SOP) হওয়া উচিত। কখনোই আর স্ট্যান্ডার্ড স্টোরেজ পারমিশন দেবেন না।

একটি অ্যাপ্লিকেশনকে ট্রিক (trick) করতে এবং এর অ্যাক্সেস কোয়ারেন্টাইন (quarantine) করতে এই সুনির্দিষ্ট স্টেপগুলো অনুসরণ করুন:

১. **App Info স্ক্রিন ওপেন করুন:** আপনার হোম স্ক্রিন বা অ্যাপ ড্রয়ারে থাকা অ্যাপ্লিকেশন আইকনে লং-প্রেস (long-press) করুন এবং **App info** ("i" আইকন) ট্যাপ করুন। বিকল্পভাবে, **Settings > Apps**-এ নেভিগেট করুন এবং অ্যাপ্লিকেশনটি সিলেক্ট করুন। ২. **অ্যাক্সেস পারমিশন: Permissions**-এ ট্যাপ করুন। ৩. **স্টোরেজ অ্যাক্সেস লোকেট করুন: Photos and videos** বা **Files and media**-তে ট্যাপ করুন (অ্যাপটির টার্গেট করা Android ভার্সনের ওপর ভিত্তি করে সঠিক শব্দগুলো পরিবর্তিত হতে পারে)। ৪. **স্ট্যান্ডার্ড অ্যাক্সেস ডিনাই করুন:** "Allow" সেকশনের অধীনে, **Don't allow** সিলেক্ট করুন। আপনাকে অবশ্যই আগে স্ট্যান্ডার্ড Android পারমিশনটি এক্সপ্লিসিটলি (explicitly) ডিনাই করতে হবে। ৫. **Storage Scopes এনগেজ করুন:** পারমিশন স্ক্রিনের একেবারে ওপরে, আপনি **Storage Scopes** লেবেলযুক্ত একটি ইউনিক GrapheneOS ব্যানার দেখতে পাবেন। এটি এনাবল করতে টগলটিতে (toggle) ট্যাপ করুন। ৬. **স্পেসিফিক স্কোপ অ্যাড করুন (ঐচ্ছিক):** Storage Scopes টগল অন (toggled on) হয়ে গেলে, **Add storage scope** বাটনে ("+" আইকন)

ট্যাপ করুন। ৭. **ফোল্ডার বা ফাইল সিলেক্ট করুন:** নেটিভ Android ফাইল পিকারটি ওপেন হবে। আপনি অ্যাপটিকে যে নির্দিষ্ট ফোল্ডার বা ফাইলটি দেখাতে চান, সেখানে নেভিগেট করুন এবং সেটি সিলেক্ট করুন। আপনার প্রয়োজনমতো আপনি যেকোনো সংখ্যক স্পেসিফিক স্কোপ যুক্ত করতে পারেন।

অ্যাপ বিহেভিয়ারের (App Behavior) ওপর একটি নোট: Storage Scopes দ্বারা কনফাইন (confine) করা একটি অ্যাপ সবসময়ই এর নিজস্ব তৈরি করা ফাইলগুলো দেখতে ও অ্যাক্সেস করতে পারে। আপনি যদি কোনো অ্যাপ ব্যবহার করে একটি PDF ডাউনলোড করেন, তবে সেই নির্দিষ্ট অ্যাপটি পরবর্তীতে সবসময় সেই নির্দিষ্ট PDF-টি রিড (read) করতে পারবে। কোনো অ্যাপের নিজস্ব তৈরি করা ফাইলগুলোতে আপনাকে ম্যানুয়ালি স্কোপ দেওয়ার প্রয়োজন নেই। আপনি শুধুমাত্র *অন্যান্য* অ্যাপ বা সিস্টেম ক্যামেরা দ্বারা তৈরি ফাইলগুলোতে অ্যাক্সেস দেওয়ার জন্যই Storage Scopes ব্যবহার করবেন।

দ্য পাওয়ার অব কম্পার্টমেন্টালাইজেশন (The Power of Compartmentalization)

Storage Scopes ফান্ডামেন্টালি (fundamentally) অপারেটর এবং সফটওয়্যার ইন্ডাস্ট্রির মধ্যে থাকা ক্ষমতার ভারসাম্যকে বদলে দেয়। এটি সেই লেভারেজটিকে (leverage) দূর করে দেয়, যা ঐতিহাসিকভাবে অ্যাপ্লিকেশনগুলো ব্যবহারকারীদের কাছ থেকে জোড়পূর্বক ডেটা সংগ্রহ করার জন্য ব্যবহার করে এসেছে।

অ্যাগ্রেসিভ অ্যাপ্লিকেশনগুলোকে একটি ফ্যাব্রিকেটেড (fabricated), শূন্য বাস্তবতায় আবদ্ধ করার মাধ্যমে আপনি বস্তুত অ্যাটাক সারফেস রিডাকশনের (Attack Surface Reduction) কোর ফিলোসফি (core philosophy) এক্সিকিউট করছেন। আপনি আপনার ইন্টেলিজেন্সকে কম্পার্টমেন্টালাইজ (compartmentalize) করছেন এবং অ্যাডভারসারিকে (adversary) ঠিক ততটুকুই ফিড (feed) করাচ্ছেন, যতটুকু আপনি তাদের দেখাতে চান—তার একটুও বেশি নয়। একটি ডিজিটাল ফোর্ট্রেস বা দুর্গ নির্মাণের মূল নির্যাস (essence) এটিই।

2.4 Contact Scopes: সোশ্যাল অ্যাপগুলোর স্যান্ডবক্সিং

যেখানে আপনার ফাইল সিস্টেম আপনার আনস্ট্রাকচার্ড ডেটা (unstructured data) ধারণ করে, আপনার অ্যাড্রেস বুক এমন কিছু ধারণ করে যা যুক্তিযুক্তভাবে আরও বিপজ্জনক: আপনার সোশ্যাল গ্রাফ (social graph)।

আপনার কন্টাক্ট ডেটাবেজ হলো আপনার সম্পূর্ণ জীবনের একটি হাইলি স্ট্রাকচার্ড, রিলেশনাল ম্যাপ। এটিতে আপনার পরিবার, সহকর্মী, চিকিৎসা প্রদানকারী এবং সোর্সের ফোন নম্বর, ইমেইল অ্যাড্রেস, হোম অ্যাড্রেস এবং অর্গানাইজেশনাল অ্যাফিলিয়েশন (organizational affiliations) রয়েছে। যখন কোনো অ্যাডভারসারি (adversary) আপনার অ্যাড্রেস বুক অধিগত করে, তখন তারা শুধু আপনাকেই কম্প্রোমাইজ (compromise) করে না; তারা আপনার সম্পূর্ণ নেটওয়ার্ককেই কম্প্রোমাইজ করে। তারা আপনার সহযোগীদের বিরুদ্ধে হাইলি টার্গেটেড স্পেয়ার-ফিশিং (spear-phishing) ক্যাম্পেইন চালাতে পারে অথবা আপনার ফ্রিকুয়েন্ট কন্টাক্টগুলোর (frequent contacts) জিপ কোড (zip codes) অ্যানালাইজ করে আপনার ফিজিক্যাল লোকেশন (physical location) বের করতে পারে।

অ্যাডভান্সড অপারেশনাল সিকিউরিটির (Operational Security বা OpSec) জগতে, আপনার সোশ্যাল গ্রাফ রক্ষা করা সবচেয়ে গুরুত্বপূর্ণ। তবুও, কমার্শিয়াল সফটওয়্যার ইন্ডাস্ট্রি এমন একটি পারমিশন সিস্টেম (permission system) ইঞ্জিনিয়ার করেছে যা জোরপূর্বক এটি এক্সট্র্যাক্ট (extract) করার জন্য ডিজাইন করা।

Contacts পারমিশনের অল-অর-নাথিং (All-or-Nothing) থ্রেট

স্ট্যান্ডার্ড Android "Contacts" পারমিশন একটি প্রিমিটিভ, অল-অর-নাথিং (all-or-nothing) প্যারাডাইমে কাজ করে।

আপনি যখন কোনো মেসেজিং অ্যাপ্লিকেশন, রাইড-শেয়ারিং অ্যাপ, বা সোশ্যাল মিডিয়া ক্লায়েন্ট ইনস্টল করেন, তখন এটি অনিবার্যভাবেই আপনার কন্টাক্টসের অ্যাক্সেস চাইবে। এর উল্লেখিত কারণটি সাধারণত ইনোকুয়াস (innocuous) বা ফানি শোনায়: "আপনার বন্ধুদের খুঁজে পেতে সাহায্য করার জন্য।"

এর না বলা বাস্তবতা অনেক বেশি ম্যালিশিয়াস (malicious)। আপনি যখন কোনো কমার্শিয়াল অ্যাপ্লিকেশনকে Contacts পারমিশন দেন, এটি শুধু লোকালি (locally) আপনার কন্টাক্টগুলো দেখে না। অ্যাপ্লিকেশনটি তাৎক্ষণিকভাবে আপনার পুরো অ্যাড্রেস বুক কপি করে এবং প্লেইন টেক্সট (plain text) হিসেবে এর রিমোট কর্পোরেট সার্ভারগুলোতে আপলোড করে। এরপর কোম্পানিটি আপনার বন্ধুবান্ধব এবং পরিবারের "শ্যাডো প্রোফাইল" (shadow profiles) তৈরি করার জন্য এই ডেটাগুলোকে এগ্রিগেট (aggregate) করে—এমনকি যদি আপনার বন্ধু এবং পরিবার কখনও নিজেরা অ্যাপটি ইনস্টল নাও করে থাকে।

আপনি যদি পারমিশন ডিনাই (deny) করার চেষ্টা করেন, তবে আপনি স্টোরেজ পারমিশনের দ্বারা ব্যবহৃত একই ফাজলামির মুখোমুখি হবেন। Telegram বা WhatsApp-এর মতো মেসেজিং অ্যাপগুলো চ্যাট শুরু করতে বা কন্টাক্ট নেম (contact names) প্রদর্শন করতে অস্বীকার করতে পারে, যা অ্যাপ্লিকেশনটিকে কার্যত অকেজো করে তোলে। আপনাকে আবারও ফাংশনালিটি (functionality) এবং প্রাইভেসি (privacy) এর মধ্যে একটি বেছে নিতে বাধ্য করা হয়।

GrapheneOS-এর সল্যুশন: Contact Scopes

ফাইল সিস্টেমের জন্য এটি যা করেছে, ঠিক সেভাবেই GrapheneOS অ্যাপ্লিকেশনটিকে একটি বানোয়াট রিয়েলিটি (fabricated reality) ফিড করে এই ফাজলামির সমাধান করে। এই ফিচারটি **Contact Scopes** নামে পরিচিত।

Contact Scopes ঠিক Storage Scopes-এর মত একই প্রিন্সিপল (principles) অনুযায়ী কাজ করে। আপনি যখন কোনো অ্যাপ্লিকেশনের জন্য Contact Scopes এনাবল করেন, তখন GrapheneOS অ্যাপটির Contacts পারমিশনের রিকোয়েস্টকে ইন্টারসেপ্ট (intercept) করে। এটি গাণিতিকভাবে অ্যাপ্লিকেশনটির কাছে মিথ্যা বলে, অ্যাপটিকে ইনফর্ম করে যে আপনি আপনার পুরো অ্যাড্রেস বুকে ফুল, আনরেস্ট্রিক্টেড অ্যাক্সেস (unrestricted access) দিয়েছেন।

অ্যাপ্লিকেশনটি অভিযোগ করা বন্ধ করে, নিজস্ব সিকিউরিটি চেক (security checks) বাইপাস করে এবং আপনার কন্টাক্টগুলো রিড (read) করার চেষ্টা করে। যাইহোক, যেহেতু এটি Contact Scopes স্যান্ডবক্সের (sandbox) ভেতরে আটকে আছে, এটি আপনার অ্যাড্রেস বুক দেখে এবং একটি সম্পূর্ণ খালি ডেটাবেজ খুঁজে পায়।

এরপর আপনি সিলেক্টিভলি (selectively) এই স্যান্ডবক্সের মধ্যে হোল (holes) তৈরি করতে পারেন, অ্যাপ্লিকেশনটিকে শুধুমাত্র আপনার নির্বাচিত, কিউরেটেড (curated) কন্টাক্টগুলোর লিস্ট ফিড করে যা আপনি এটিকে দেখাতে চান।

কিউরেটেড ফিড সিনারিও: Telegram এবং Signal

Telegram-এর মতো একটি মেসেজিং অ্যাপ্লিকেশনের ডিপ্লয়মেন্ট (deployment) বিবেচনা করুন। তিনজন নির্দিষ্ট প্রফেশনাল সহযোগীর সাথে যোগাযোগ করার জন্য আপনার Telegram ব্যবহার করার প্রয়োজন হতে পারে। আপনি চাইবেন না যে Telegram, বা এর প্যারেন্ট কোম্পানি, আপনার স্বামী/স্ত্রী, আপনার সন্তান বা আপনার ডাক্তারের ফোন নম্বরগুলো তাদের দখলে রাখুক।

স্ট্যান্ডার্ড Android-এ, আপনাকে সবাইকে সমর্পণ করতে হবে। GrapheneOS-এর অধীনে, আপনি একটি Contact Quarantine এক্সিকিউট (execute) করতে পারেন:

1. আপনি Telegram ইনস্টল করেন এবং স্ট্যান্ডার্ড Contacts পারমিশন ডিনাই (deny) করেন।
2. আপনি Contact Scopes এনাবল (enable) করেন।
3. আপনি আপনার মাস্টার অ্যাড্রেস বুক থেকে তিনজন নির্দিষ্ট প্রফেশনাল সহযোগীকে সিলেক্ট করেন এবং তাদের স্কোপে (scope) অ্যাড করেন।

Telegram যখন লঞ্চ হয়, এটি বিশ্বাস করে যে এটি আপনার পুরো অ্যাড্রেস বুক রিড করছে। এটি ঠিক তিনজনকে দেখতে পায়। এটি ধরে নেয় যে আপনি এমন একজন ব্যক্তি যে পুরো পৃথিবীতে মাত্র তিনজনকেই চেনে। এটি সেই তিনটি কন্টাক্টকে সিঙ্ক (syncs) করে, আপনাকে তাদের সাথে স্বাভাবিকভাবে মেসেজ করার অনুমতি দেয় এবং আপনার ডিভাইসে থাকা অন্যান্য পাঁচশো অত্যন্ত সেনসিটিভ কন্টাক্টের (highly sensitive contacts) প্রতি সম্পূর্ণ ব্লাইন্ড (blind) থাকে।

আপনার সোশ্যাল গ্রাফ সুরক্ষিত থাকে, এবং অ্যাপ্লিকেশনটি নিখুঁতভাবে ফাংশন (functions) করে।

দ্য ইমিউটেবল ডেটাবেজ (Immutable Database): Read-Only এনফোর্সমেন্ট

Contact Scopes সিকিউরিটির আরেকটি অত্যন্ত ক্রিটিক্যাল লেয়ার (critical layer) প্রদান করে: **ইমিউটেবিলিটি** (**immutability**)।

যখন কোনো অ্যাপ্লিকেশনকে স্ট্যান্ডার্ড Android Contacts পারমিশন দেওয়া হয়, তখন এটি আপনার কন্টাক্টগুলো রিড করার ক্ষমতা পায়, কিন্তু একই সাথে সেগুলোকে *রাইট* (*write*) এবং *মডিফাই* (*modify*) করারও ক্ষমতা অর্জন করে। এটি একটি বিশাল সিকিউরিটি ভালনারেবিলিটি (security vulnerability) তৈরি করে। একটি কম্প্রোমাইজড বা পোরলি কোডেড অ্যাপ্লিকেশন নীরবে কন্টাক্ট মুছে ফেলতে পারে, কোনো হোস্টাইল অ্যাক্টরের (hostile actor) কাছে আপনার কল রিডাইরেক্ট (redirect) করার জন্য ফোন নম্বর পরিবর্তন করতে পারে, বা আপনার ডেটাবেজকে স্প্যাম (spam) এন্ট্রি দিয়ে আবর্জনায় পূর্ণ করতে পারে।

যখন Contact Scopes অ্যাক্টিভ থাকে, তখন স্যান্ডবক্সটি কঠোরভাবে একটি রিড-ওনলি পলিসি (read-only policy) এনফোর্স করে। অ্যাপ্লিকেশনটিকে আপনার মাস্টার অ্যাড্রেস বুকের কোনো এন্ট্রি মডিফাই করা, যোগ করা বা ডিলিট করা থেকে সম্পূর্ণরূপে বার (barred) করা হয়। এমনকি যেসব কন্টাক্ট আপনি এক্সপ্লিসিটলি (explicitly) দেখার অনুমতি দিয়েছেন, সেগুলোও অ্যাপটি দ্বারা পরিবর্তন করা সম্ভব নয়।

যদি আপনার মেসেজিং অ্যাপে একটি নতুন কন্টাক্ট যোগ করার প্রয়োজন হয়, আপনি শুধুমাত্র আপনার প্রধান GrapheneOS Contacts অ্যাপে কন্টাক্টটি যোগ করবেন, এবং এরপর সেই নতুন এন্ট্রিটি অ্যাপ্লিকেশনটির Contact Scope-এ যোগ করবেন। মাস্টার ডেটাবেজটি অক্ষত (pristine) থাকে এবং সম্পূর্ণভাবে আপনার ডিরেক্ট কন্ট্রোলে (direct control) থাকে।

Contact Scopes কীভাবে কনফিগার করবেন

Contact Scopes ডিপ্লয় করাটা Storage Scopes ডিপ্লয় করার মতোই। যখনই কোনো অ্যাপ্লিকেশন আপনার সোশ্যাল গ্রাফে অ্যাক্সেস চাইবে, তখন এটি আপনার ডিফল্ট অ্যাকশন (default action) হয়ে উঠতে হবে।

একটি অ্যাপ্লিকেশনকে ধোঁকা দিতে এবং আপনার নেটওয়ার্ককে কোয়ারেন্টাইন (quarantine) করতে এই সুনির্দিষ্ট স্টেপগুলো অনুসরণ করুন:

1. **App Info স্ক্রিন ওপেন করুন:** আপনার হোম স্ক্রিন বা অ্যাপ ড্রয়ারে অ্যাপ্লিকেশন আইকনটির ওপর লং-প্রেস (Long-press) করুন এবং **App info** ("i" আইকন) ট্যাপ করুন। বিকল্পভাবে, **Settings > Apps**-এ নেভিগেট (navigate) করুন এবং অ্যাপ্লিকেশনটি সিলেক্ট করুন।
2. **পারমিশনে অ্যাক্সেস করুন: Permissions**-এ ট্যাপ করুন।
3. **Contacts অ্যাক্সেস লোকেট করুন: Contacts**-এ ট্যাপ করুন।
4. **স্ট্যান্ডার্ড অ্যাক্সেস ডিনাই করুন:** "Allow" সেকশনের অধীনে, **Don't allow** সিলেক্ট করুন। এক্সটর্শন বাইপাস (extortion bypass) ট্রিগার (trigger) করার জন্য আপনাকে অবশ্যই প্রথমে স্ট্যান্ডার্ড Android পারমিশন এক্সপ্লিসিটলি (explicitly) ডিনাই করতে হবে।
5. **Contact Scopes এনগেজ করুন:** পারমিশন স্ক্রিনের একদম ওপরে, বানোয়াট স্যান্ডবক্সটি এনাবল করতে **Contact Scopes**-এর টগলটিতে ট্যাপ করুন।
6. **স্পেসিফিক স্কোপস অ্যাড করুন:** একবার Contact Scopes অ্যাক্টিভ হয়ে গেলে, **Add contact** বা **Add group** বোতামে ("+" আইকন) ট্যাপ করুন।
7. **আপনার টার্গেট সিলেক্ট করুন:** আপনার মাস্টার অ্যাড্রেস বুকটি ওপেন হবে। নির্দিষ্ট ব্যক্তিদের সিলেক্ট করুন যাদের আপনি অ্যাপ্লিকেশনটির কাছে রিভিল (reveal) করতে চান।

সোশ্যাল গ্রাফের কম্পার্টমেন্টালাইজিং (Compartmentalizing)

আধুনিক ডিজিটাল ব্যাটলফিল্ডে (battlefield), ডেটা শুধুমাত্র চুরিই হয় না; এটি কোরিলেট (correlate) করা হয়। অ্যাডভারসারিরা (Adversaries) কেবল আপনি কী বলেন তা থেকেই নয়, বরং আপনি কাকে চেনেন তা থেকেও কম্প্রিহেন্সিভ প্রোফাইল (comprehensive profiles) তৈরি করে।

Contact Scopes ডিপ্লয় করার মাধ্যমে, আপনি সেই কানেক্টিভ টিস্যুটি (connective tissue) ছিন্ন করেন যার ওপর ডেটা ব্রোকাররা (data brokers) নির্ভর করে। আপনি স্ট্রিক্ট কম্পার্টমেন্টালাইজেশন (strict compartmentalization) এনফোর্স করেন, এটি নিশ্চিত করে যে ব্যবসার জন্য ব্যবহৃত একটি মেসেজিং অ্যাপ ব্যক্তিগত যোগাযোগের জন্য ব্যবহৃত অন্য একটি

অ্যাপের কন্টাক্টগুলোকে ক্রস-রেফারেন্স (cross-reference) করতে পারবে না। আপনি আপনার সোশ্যাল গ্রাফের ওপর অ্যাবসলিউট অপারেশনাল কন্ট্রোল (absolute operational control) ধরে রাখেন, যা নিখুঁত ইউজেবিলিটি (usability) বজায় রেখে সার্ভেইল্যান্স ইকোনমিকে (surveillance economy) এর সবচেয়ে মূল্যবান রিসোর্স থেকে বঞ্চিত করে।

ফোর্ট্রেস (fortress) অটুট আছে। এখন, আমাদের নেটওয়ার্কের দিকেই নজর দিতে হবে।

2.5 দ্য আল্টিমেট কিল সুইচ (The Ultimate Kill Switch): পার-অ্যাপ (Per-App) নেটওয়ার্ক টগলস

সাইবার ওয়ারফেয়ারের (cyber warfare) ক্ষেত্রে, একটি ডেটা ব্রিচ (data breach) দুটি স্বতন্ত্র পর্যায়ে ঘটে: কালেকশন (collection) এবং এক্সফিলট্রেশন (exfiltration)।

প্রথমত, হোস্টাইল (hostile) সফটওয়্যারটিকে অবশ্যই টার্গেট ডেটা কালেক্ট করতে হবে (যেমন, আপনার কিস্ট্রোক (keystrokes) লগ করা, আপনার ছবিগুলো ইনডেক্স করা, বা আপনার মাইক্রোফোন রেকর্ড করা)। দ্বিতীয়ত, এবং সবচেয়ে গুরুত্বপূর্ণভাবে, সফটওয়্যারটিকে অবশ্যই সেই ডেটা *এক্সফিলট্রেট* (*exfiltrate*) করতে হবে। এটিকে অবশ্যই একটি নেটওয়ার্ক কানেকশন ওপেন করতে হবে এবং সংগৃহীত ইন্টেলিজেন্স (intelligence) অ্যাডভারসারির (adversary) কমান্ড-অ্যান্ড-কন্ট্রোল সার্ভারে (command-and-control server) ট্রান্সমিট (transmit) করতে হবে।

আপনি যদি নেটওয়ার্ক কানেকশনটি ছিন্ন করতে পারেন, তবে আপনি থ্রেটটিকে (threat) নিউট্রালাইজ (neutralize) করে ফেলবেন। কোনো ম্যালিশিয়াস (malicious) অ্যাপ্লিকেশন আপনার প্রতিটি শব্দ সফলভাবে রেকর্ড করলেও তাতে কিছু যায় আসে না; যদি এটি অডিও ফাইলটি আপনার ডিভাইসের বাইরে ট্রান্সমিট করতে না পারে, তবে সেই ইন্টেলিজেন্স অ্যাটাকারের (attacker) জন্য সম্পূর্ণ অকেজো।

আশ্চর্যজনকভাবে, স্ট্যান্ডার্ড কমার্শিয়াল অপারেটিং সিস্টেমগুলো ব্যবহারকারীকে এই ডিফেন্সিভ ক্যাপাবিলিটি (defensive capability) থেকে পুরোপুরি বঞ্চিত করে।

Android-এর ব্লাইন্ড স্পট (Blind Spot)

একটি স্ট্যান্ডার্ড Android ডিভাইসে, আপনার ক্যামেরা, আপনার মাইক্রোফোন এবং আপনার লোকেশনের ওপর আপনার গ্র্যানুলার কন্ট্রোল (granular control) রয়েছে। তবে, সবচেয়ে বিপজ্জনক পারমিশনটির ওপর আপনার কোনোই নিয়ন্ত্রণ নেই: **ইন্টারনেট অ্যাক্সেস (Internet Access)**।

আপনি যখন Google Play Store থেকে কোনো অ্যাপ্লিকেশন ইনস্টল করেন, তখন `INTERNET` পারমিশনটি স্বয়ংক্রিয়ভাবে, নীরবে এবং স্থায়ীভাবে গ্র্যান্ট (grant) করা হয়। আপনি এটি রিভোক (revoke) করতে পারবেন না। আপনি এটি টগল অফ (toggle off) করতে পারবেন না। অপারেটিং সিস্টেমটি ইন্টারনেটের একটি স্থায়ী, ওপেন পাইপলাইনের (open pipeline) মাধ্যমে প্রতিটি সিঙ্গেল অ্যাপ্লিকেশনকে ইমপ্লিসিটলি (implicitly) বিশ্বাস করে।

এই আর্কিটেকচারাল ফ্ল'র (architectural flaw) কারণে সাধারণ ক্যালকুলেটর বা ফ্ল্যাশলাইট অ্যাপের মতো "অফলাইন" অ্যাপ্লিকেশনগুলো ব্যাকগ্রাউন্ডে নীরবে আপনার ইউজেস টেলিমেট্রি (usage telemetry), আপনার IP অ্যাড্রেস এবং আপনার ডিভাইস আইডেন্টিফায়ার (device identifiers) হারভেস্ট (harvest) করতে পারে এবং সেগুলো কর্পোরেট ডেটা ব্রোকারদের (corporate data brokers) কাছে ট্রান্সমিট করতে পারে।

GrapheneOS মাস্টার সুইচ (Master Switch)

GrapheneOS ফান্ডামেন্টালি (fundamentally) এই আর্কিটেকচারকে প্রত্যাখ্যান করে। এটি একটি প্রোপাইটারি (proprietary), OS-লেভেল কিল সুইচ (kill switch) ইন্ট্রোডিউস (introduce) করে: **নেটওয়ার্ক পারমিশন টগল** (**The Network Permission Toggle**)।

সফটওয়্যার-বেইজড ফায়ারওয়াল (firewalls) বা VPN রাউটিং ট্রিকগুলোর (VPN routing tricks) (যাকে প্রায়শই সফিস্টিকেটেড ম্যালওয়্যার দ্বারা বাইপাস করা সম্ভব) মতো নয়, GrapheneOS নেটওয়ার্ক টগলটি অপারেটিং সিস্টেমের গভীরতম স্তরে কাজ করে। এটি সমস্ত এভেইলেবল নেটওয়ার্কে (Wi-Fi, Cellular এবং এমনকি ডিভাইস-লোকাল localhost রাউটিং) ডিরেক্ট এবং ইনডিরেক্ট উভয় অ্যাক্সেসকেই সম্পূর্ণভাবে ছিন্ন করে।

আপনি যখন কোনো অ্যাপ্লিকেশনের জন্য নেটওয়ার্ক পারমিশন রিভোক করেন, GrapheneOS তখন একটি চমৎকার ডিসেপশন (deception) এক্সিকিউট (execute) করে। কানেকশনটি ফোর্সফুলি ব্লক করে অ্যাপ্লিকেশনটিকে ক্র্যাশ করানোর পরিবর্তে, OS কেবল অ্যাপটির সাথে মিথ্যা বলে। এটি স্ট্যান্ডার্ড সিস্টেম এরর (system errors) রিটার্ন করে যা ইঙ্গিত দেয় যে ডিভাইসটি সাময়িকভাবে অফলাইনে বা এয়ারপ্লেন মোডে (airplane mode) রয়েছে। অ্যাপ্লিকেশনটি চমৎকারভাবে এররটি হ্যান্ডেল করে, নেটওয়ার্ক "ফিরে আসার" জন্য অপেক্ষা করে এবং লোকালি ফাংশন করা চালিয়ে যায়।

ট্যাকটিকাল ডিপ্লয়মেন্টস (Tactical Deployments)

কোনো অ্যাপ্লিকেশনের নেটওয়ার্ক অ্যাক্সেস ছিন্ন করার ক্ষমতা একটি গভীর ট্যাকটিকাল অ্যাডভান্টেজ (tactical advantage) তৈরি করে। এটি একজন এলিট অপারেটরকে (elite operator) একটি হারমেটিক্যালি সিল্ড (hermetically sealed) এনভায়রনমেন্টের (environment) ভেতরে নিরাপদে হোস্টাইল, প্রোপাইটারি বা আনট্রাস্টেড (untrusted) সফটওয়্যার ব্যবহার করার অনুমতি দেয়।

নিম্নলিখিত হাই-ভ্যালু (high-value) ডিপ্লয়মেন্টগুলো বিবেচনা করুন:

১. প্রোপাইটারি কীবোর্ড (Gboard) আপনার কীবোর্ড আক্ষরিক অর্থেই একটি কীলগার (keylogger)। আপনার টাইপ করা প্রতিটি পাসওয়ার্ড, সার্চ কোয়েরি (search query) এবং ব্যক্তিগত মেসেজ কীবোর্ড অ্যাপ্লিকেশন দ্বারা প্রসেস করা হয়। অনেক ইউজার Google-এর "Gboard" বা Microsoft-এর "SwiftKey"-এর মতো প্রোপাইটারি কীবোর্ডগুলোর প্রেডিক্টিভ টাইপিং (predictive typing) এবং সোয়াইপ (swipe) ফিচারগুলো পছন্দ করেন, কিন্তু এগুলো ব্যবহার করতে অস্বীকার করেন কারণ এই কীবোর্ডগুলো কুখ্যাতভাবে তাদের কর্পোরেট সার্ভারগুলোতে টাইপিং টেলিমেট্রি ট্রান্সমিট করে। *এর সল্যুশন:* Gboard ইনস্টল করুন, অবিলম্বে এর নেটওয়ার্ক পারমিশন রিভোক করুন এবং এটি নিরাপদে ব্যবহার করুন। এটি একটি লোকাল কীবোর্ড হিসেবে পুরোপুরি ফাংশন করবে, কিন্তু আপনার কিস্ট্রোকগুলো এক্সফিলট্রেট করার ক্ষমতা গাণিতিকভাবে নিউট্রালাইজড (neutralized) হয়ে যাবে।

২. স্কেচি (Sketchy) অফলাইন গেম আপনি বা আপনার পরিবারের কোনো সদস্য হয়তো একটি জনপ্রিয় মোবাইল গেম ইনস্টল করতে চাইতে পারেন। এই গেমগুলো প্রায়শই পুরোপুরি অফলাইনে খেলা যায়, তবে এগুলো অ্যাগ্রেসিভ ট্র্যাকিং SDKs (aggressive tracking SDKs) এবং টার্গেটেড অ্যাডভারটাইজমেন্ট (targeted advertisements) দ্বারা পরিপূর্ণ থাকে, যা আপনার ব্যাটারি ড্রেইন (drain) করে এবং নেটওয়ার্কের সাথে কানেক্টেড থাকলে আপনার ডেটা হারভেস্ট করে। *এর সল্যুশন:* নেটওয়ার্ক পারমিশন রিভোক করুন। গেমটি আর অ্যাডভারটাইজমেন্ট ডাউনলোড করতে পারবে না এবং এটি আপনার ইউজেস স্ট্যাটিস্টিকস (usage statistics) আপলোড করতে পারবে না। আপনার জন্য একটি স্টেরাইল (sterile), অফলাইন গেমিং এক্সপেরিয়েন্স অবশিষ্ট থাকবে।

৩. লোকাল গ্যালারি অ্যাপ (Local Gallery App) আপনার মিডিয়া অর্গানাইজ করার জন্য হয়তো আপনি একটি নির্দিষ্ট থার্ড-পার্টি (third-party) ফটো গ্যালারি অ্যাপ পছন্দ করেন, কিন্তু আপনি কোনো বাগ (bug) বা সাইলেন্ট আপডেটের (silent update) কারণে অ্যাপটি স্বয়ংক্রিয়ভাবে আপনার ব্যক্তিগত ছবিগুলো কোনো র‍্যান্ডম ক্লাউড সার্ভারে (cloud server) আপলোড করার ঝুঁকি নিতে চান না। *এর সল্যুশন:* নেটওয়ার্ক পারমিশন রিভোক করুন। একটি ফটো গ্যালারির ইন্টারনেট অ্যাক্সেস করার কোনো যৌক্তিক কারণ নেই। কানেকশনটি ছিন্ন করার মাধ্যমে, আপনি গ্যারান্টি দিচ্ছেন যে আপনার মিডিয়া আপনার ডিভাইসের ফিজিক্যাল সিলিকনের (physical silicon) ওপর পার্মানেন্টলি কোয়ারেন্টাইন্ড (permanently quarantined) থাকবে।

কিল সুইচ এক্সিকিউট (Executing) করা

যেহেতু GrapheneOS এই ফিচারের ক্রিটিক্যাল ইম্পর্ট্যান্স (critical importance) বুঝতে পারে, তাই আপনি যখন কোনো অ্যাপ্লিকেশন ইনস্টল করেন ঠিক সেই মুহূর্তেই এটি আপনার কাছে নেটওয়ার্ক টগলটি উপস্থাপন করে। যাইহোক, আপনি যেকোনো সময় যেকোনো ইনস্টল করা অ্যাপের জন্য এই পারমিশন অডিট (audit) এবং রিভোক করতে পারেন।

১. **App Info স্ক্রিন ওপেন করুন:** অ্যাপ্লিকেশন আইকনটির ওপর লং-প্রেস করুন এবং **App info** ট্যাপ করুন, অথবা **Settings > Apps**-এ নেভিগেট করুন। ২. **পারমিশনে অ্যাক্সেস করুন: Permissions**-এ ট্যাপ করুন। ৩. **নেটওয়ার্ক অ্যাক্সেস লোকেট করুন:** স্ক্রল ডাউন করে **Network**-এ ট্যাপ করুন। ৪. **সুইচটি এক্সিকিউট করুন: Don't allow** সিলেক্ট করুন।

নেটওয়ার্ক টগলে মাস্টারিং (mastering) করার মাধ্যমে, আপনি আল্টিমেট ডিফেন্সিভ পোসচার (ultimate defensive posture) অর্জন করেন: **এক্সফিলট্রেশন নিউট্রালাইজেশন** (**Exfiltration Neutralization**)। আপনি যে সফটওয়্যারগুলো ইনস্টল করেন তার ডেভেলপারদের বিশ্বাস করতে আপনি আর বাধ্য নন। আপনাকে তাদের প্রাইভেসি পলিসি (privacy policies) পড়তে হবে না এবং আপনাকে তাদের প্রতিশ্রুতির ওপর নির্ভর করতে হবে না। তাদের কমিউনিকেশন লাইনগুলো শারীরিকভাবে ছিন্ন করার মাধ্যমে, আপনি ট্রাস্টের (trust) পরিবর্তে গণিত দিয়ে আপনার প্রাইভেসি এনফোর্স করেন।

2.6 এক্সট্রিম কম্পার্টমেন্টালাইজেশন (Extreme Compartmentalization): ইউজার প্রোফাইলস (User Profiles)

নেভাল আর্কিটেকচারে (naval architecture), যখন ইঞ্জিনিয়াররা কোনো যুদ্ধজাহাজের ডিজাইন করেন, তখন তারা একটি একক, বিশাল, ফাঁপা হাল (hollow hull) তৈরি করেন না। যদি একটি ফাঁপা জাহাজে টর্পেডোর (torpedo) আঘাত লাগে, তবে বিনা বাধায় ভেতরে পানি ঢুকে যায় এবং পুরো জাহাজটি ডুবে যায়। এর পরিবর্তে, ইঞ্জিনিয়াররা হালটিকে ডজন ডজন ওয়াটারটাইট (watertight), ফিজিক্যালি আইসোলেটেড কম্পার্টমেন্টে (physically isolated compartments) বিভক্ত করেন, যা ভারী স্টিল বাল্কহেড (steel bulkheads) দ্বারা আলাদা করা থাকে। যদি ৩ নম্বর কম্পার্টমেন্টে টর্পেডোর আঘাত লাগে, তবে বাল্কহেডগুলো সিল (sealed) করে দেওয়া হয়। ৩ নম্বর কম্পার্টমেন্টটি পানিতে প্লাবিত হয়, কিন্তু জাহাজের বাকি অংশটি পুরোপুরি ভাসমান থাকে।

এটিই হলো **কম্পার্টমেন্টালাইজেশন**-এর (**Compartmentalization**) ফান্ডামেন্টাল প্রিন্সিপল (foundational principle)। নেভাল ওয়ারফেয়ার (naval warfare) এবং ইন্টেলিজেন্স অপারেশন (intelligence operations) উভয়ের জন্যই এটি সবচেয়ে ক্রিটিক্যাল ডকট্রিন (critical doctrine)। আপনি কখনোই আপনার সমস্ত ইন্টেলিজেন্স, সম্পদ, বা ভালনারেবিলিটি (vulnerabilities) একই রুমে রাখবেন না। যদি একজন একক অপারেটিভ (operative) কম্প্রোমাইজড (compromised) হয়ে যায়, তবে কোনোভাবেই যেন পুরো নেটওয়ার্কটি কলাপ্স (collapse) না করে।

ডিজিটাল জগতেও একটি স্ট্যান্ডার্ড কমার্শিয়াল স্মার্টফোন হলো একটি ফাঁপা জাহাজ। আপনার ব্যাংকিং অ্যাপ্লিকেশন, আপনার এনক্রিপ্টেড মেসেজিং অ্যাপ্লিকেশন, আপনার ওয়েব ব্রাউজার এবং আপনার ডাউনলোড করা সেই স্কেচি মোবাইল গেমটি—সবগুলোই ঠিক একই হালের মধ্যে ভাসছে। যদি কোনো সফিস্টিকেটেড অ্যাডভারসারি (sophisticated adversary) লোকাল কোড এক্সিকিউশন (local code execution) গেইন (gain) করার জন্য আপনার ওয়েব ব্রাউজারে একটি জিরো-ডে এক্সপ্লয়েট (zero-day exploit) ব্যবহার করে, তবে তাৎক্ষণিকভাবে তারা পুরো জাহাজটিতেই অ্যাক্সেস পেয়ে যায়।

GrapheneOS আপনাকে সেই বাল্কহেডগুলো তৈরি করার সুযোগ দেয়। **ইউজার প্রোফাইলস**-এর (**User Profiles**) ট্যাকটিকাল ডিপ্লয়মেন্টের (tactical deployment) মাধ্যমে এটি অর্জন করা হয়।

ইউজার প্রোফাইলস-এর ক্রিপ্টোগ্রাফিক রিয়েলিটি (Cryptographic Reality)

স্ট্যান্ডার্ড Android সর্বোচ্চ চারটি ইউজার প্রোফাইল সাপোর্ট করে। GrapheneOS এই লিমিটেশনটি ভেঙে ফেলে, যা অপারেটরদের **৩২টি সম্পূর্ণ আইসোলেটেড ইউজার প্রোফাইল** (৩১ জন সেকেন্ডারি ইউজার এবং একটি গেস্ট প্রোফাইল) আর্কিটেক্ট (architect) করার অনুমতি দেয়।

GrapheneOS-এ ইউজার প্রোফাইলগুলো কেবল কোনো কসমেটিক (cosmetic) ইউজার ইন্টারফেসের কৌশল নয়। এগুলো গাণিতিকভাবে আইসোলেটেড, ক্রিপ্টোগ্রাফিক্যালি ডিস্টিংক্ট এনভায়রনমেন্ট (cryptographically distinct environments)।

আপনি যখন একটি নতুন ইউজার প্রোফাইল তৈরি করেন, তখন GrapheneOS ফিজিক্যাল স্টোরেজের একটি অংশকে পার্টিশন (partitions) করে। এটি সেই নির্দিষ্ট প্রোফাইলের জন্য ফাইল-বেইজড এনক্রিপশন কি'র (file-based encryption keys) সম্পূর্ণ নতুন এবং ইউনিক (unique) একটি সেট জেনারেট (generate) করে, যা প্রোফাইলটিতে অ্যাসাইন (assign) করা ইউনিক পিন (PIN) বা পাসওয়ার্ড থেকে ডিরাইভ (derive) করা হয়।

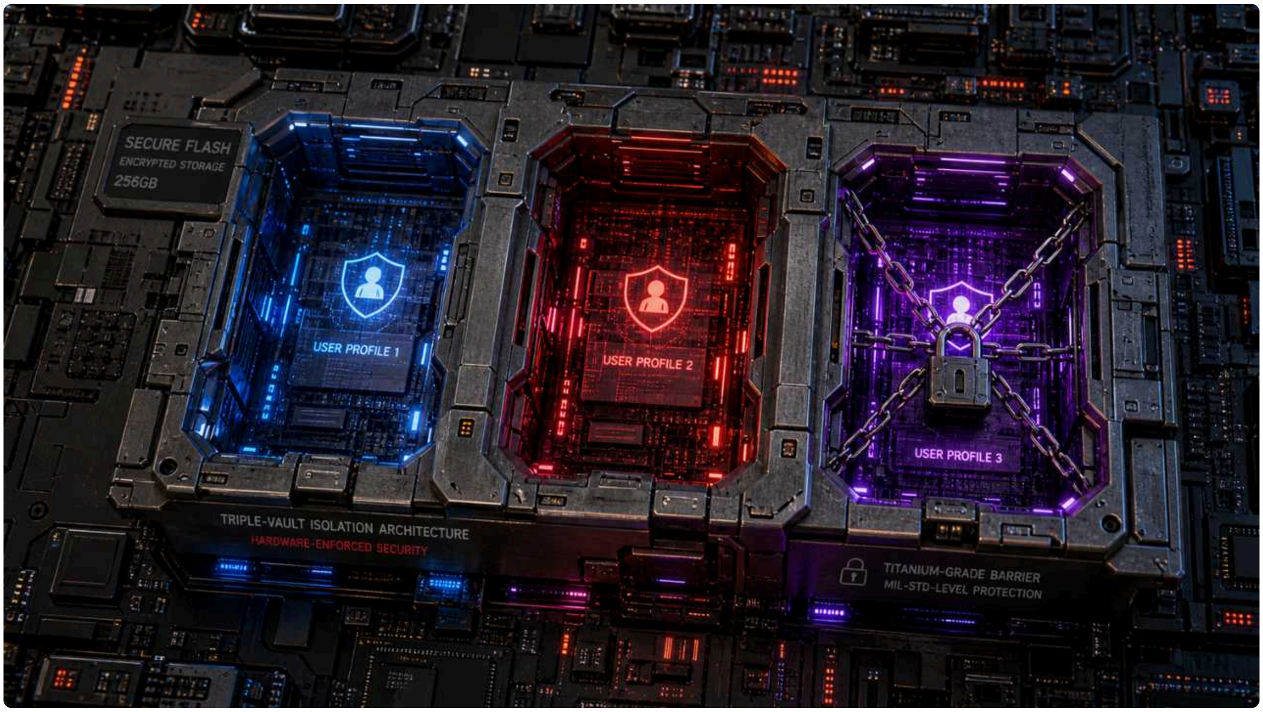
এক্ষেত্রে টেকনিক্যাল সেপারেশনটি (technical separation) হলো অ্যাবসলিউট (absolute):

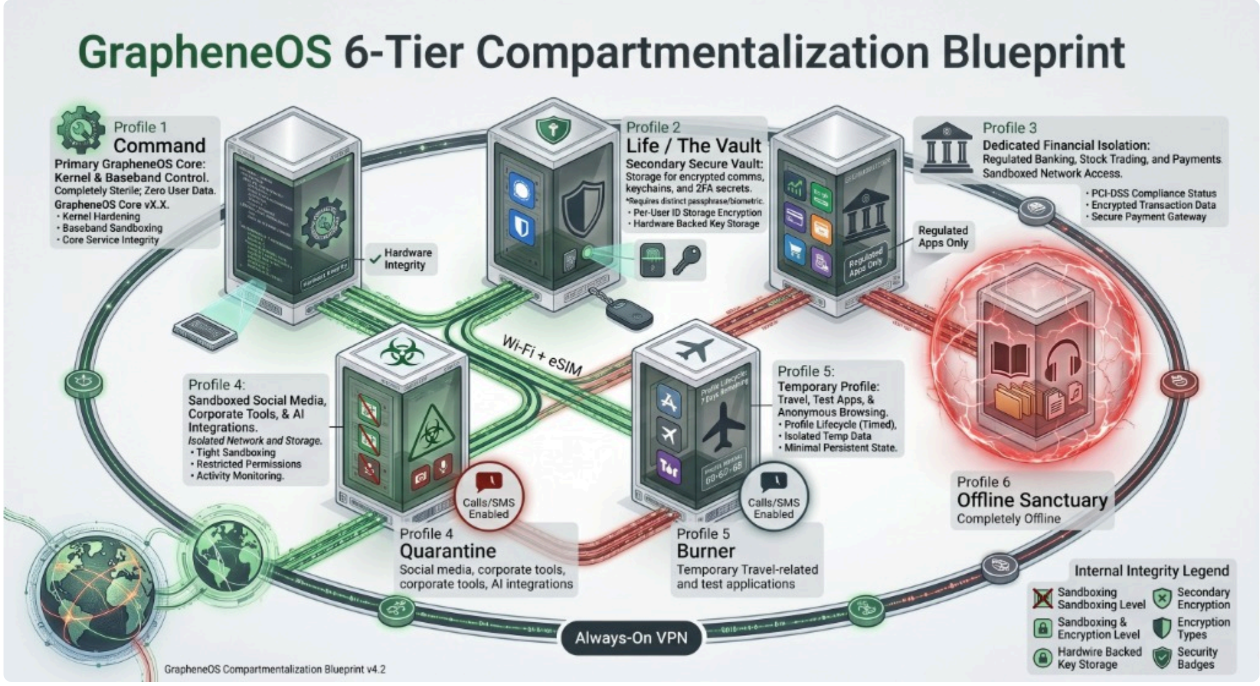
- **ডেটা আইসোলেশন (Data Isolation):** প্রোফাইল এ (Profile A)-এর ফাইল সিস্টেমটি প্রোফাইল বি (Profile B)-এর কাছে সম্পূর্ণ অদৃশ্য এবং গাণিতিকভাবে ইনঅ্যাক্সেসিবল (inaccessible)।
- **অ্যাপ আইসোলেশন (App Isolation):** প্রোফাইল এ-তে ইনস্টল করা অ্যাপ্লিকেশনগুলো প্রোফাইল বি-তে ইনস্টল করা অ্যাপ্লিকেশনগুলোকে দেখতে, স্ক্যান করতে, বা সেগুলোর সাথে ইন্টার‍্যাক্ট (interact) করতে পারে না। তারা কোনো অ্যাপ্লিকেশন স্যান্ডবক্স শেয়ার করে না।
- **কি পার্জিং (Key Purging):** GrapheneOS-এ একটি "End Session" ফাংশন রয়েছে। আপনি যখন কোনো প্রোফাইল থেকে লগ আউট (log out) করেন (কেবলমাত্র এটি থেকে সুইচ অ্যাওয়ে বা switch away করার পরিবর্তে), তখন অপারেটিং সিস্টেমটি অ্যাগ্রেসিভলি (aggressively) সেই প্রোফাইলের এনক্রিপশন কি'গুলোকে হার্ডওয়্যার রেজিস্টার (hardware registers) এবং সিস্টেম মেমোরি (system memory) থেকে পার্জ (purges) করে দেয়। সেই প্রোফাইলের ডেটা তাৎক্ষণিকভাবে "অ্যাট রেস্ট" ("at rest") অবস্থায় চলে যায় এবং প্রোফাইলটির জন্য নির্দিষ্ট পিনটি পুনরায় এন্টার (re-entering) করা ছাড়া সেটি ডিক্রিপ্ট (decrypt) করা সম্ভব নয়।

যদি কোনো স্টেট-স্পন্সরড অ্যাডভারসারি (state-sponsored adversary) সফলভাবে প্রোফাইল বি-এর কোনো অ্যাপ্লিকেশনে একটি হাইলি টার্গেটেড ম্যালওয়্যার পেলোড (highly targeted malware payload) ডেলিভার (deliver) করতে সক্ষম হয়, তবে ম্যালওয়্যারটি সেখানেই ট্র্যাপড (trapped) হয়ে যায়। এটি বাল্কহেড ট্রাভার্স (traverse) করে এনক্রিপশন কি'গুলো চুরি করতে পারে না বা প্রোফাইল এ-তে থাকা সিকিউর কমিউনিকেশনে (secure communications) অ্যাক্সেস করতে পারে না।

দ্য প্রোফাইল ব্লুপ্রিন্ট (The Profile Blueprint): আর্কিটেকটিং দ্য ফোর্ট্রেস (Architecting the Fortress)

৩২টি প্রোফাইল তৈরি করার ক্ষমতা থাকার অর্থ এই নয় যে আপনার অ্যাপ্লিকেশনগুলোকে এলোমেলোভাবে ছড়িয়ে ছিটিয়ে রাখা উচিত। এক্সট্রিম কম্পার্টমেন্টালাইজেশনের জন্য ডেলিবরেট আর্কিটেকচার (deliberate architecture) প্রয়োজন। আপনাকে অবশ্যই আপনার থ্রেট মডেলটি (threat model) ইভ্যালুয়েট (evaluate) করতে হবে এবং অ্যাপ্লিকেশনগুলোর ট্রাস্ট লেভেল (trust level) ও পারপাসের (purpose) ওপর ভিত্তি করে সেগুলোকে গ্রুপ করতে হবে।





একজন হাই-সিকিউরিটি অপারেটরের জন্য, আমরা নিচের **প্রোফাইল ব্লুপ্রিন্ট (Profile Blueprint)** রেকমেন্ড (recommend) করি।

প্রোফাইল ১: মূল প্রোফাইল (The Owner) বা সিস্টেম কমান্ড (System Command) ডিভাইসটি প্রথমবার বুট (boot) করার সময় যে প্রোফাইলটি তৈরি হয়, সেটিই হলো মূল প্রোফাইল (Owner profile)। এই প্রোফাইলটির কাছে বিশেষ অ্যাডমিনিস্ট্রেটিভ প্রিভিলেজ (administrative privileges) থাকে, যা সেকেন্ডারি প্রোফাইলগুলোর (secondary profiles) থাকে না; যেমন—পুরো সিস্টেমব্যাপী Wi-Fi কানেকশন ম্যানেজ করা, অপারেটিং সিস্টেম আপডেট ইনস্টল করা এবং গ্লোবাল সেটিংস (global settings) ডিক্টেট (dictate) করা।

যেহেতু এটি পুরো সিস্টেমের নিয়ন্ত্রণ রাখে, তাই মূল প্রোফাইলটিকে একটি স্টেরাইল কমান্ড সেন্টার (sterile command center) হিসেবে বিবেচনা করতে হবে।

- **দ্য রুল (The Rule):** আপনি কখনোই মূল প্রোফাইলে থার্ড-পার্টি অ্যাপ্লিকেশন, মেসেজিং অ্যাপ বা ওয়েব ব্রাউজার ইনস্টল করবেন না। এটি সম্পূর্ণরূপে পার্সোনাল ডেটা (personal data) থেকে মুক্ত থাকা উচিত। আপনি শুধুমাত্র অপারেটিং সিস্টেম আপডেট করতে, সিস্টেম সেটিংস কনফিগার করতে অথবা নতুন সেকেন্ডারি প্রোফাইল তৈরি করার জন্যই মূল প্রোফাইলে লগ-ইন করবেন। অ্যাডমিনিস্ট্রেটিভ কাজগুলো সম্পন্ন হওয়ার পর, আপনাকে তাৎক্ষণিকভাবে সেশন এন্ড (End Session) করে আপনার ডেইলি প্রোফাইলে (daily profile) সুইচ করতে হবে।

> 💡 ***Pro Tip:*** *সিস্টেম অ্যাডমিনিস্ট্রেশনের কাজ শেষ করার পর, মূল প্রোফাইল থেকে বের হওয়ার সময় সবসময় "End Session" ব্যবহার করবেন (শুধুমাত্র "Switch User" নয়)। এটি RAM থেকে মূল প্রোফাইলের এনক্রিপশন কি'গুলোকে (encryption keys) পার্জ (purge) করে দেয়, যা নিশ্চিত করে যে একটি কম্প্রোমাইজড সেকেন্ডারি প্রোফাইলও আপনার অ্যাডমিনিস্ট্রেটিভ ক্রেডেনশিয়ালসে (administrative credentials) অ্যাক্সেস করতে পারবে না।*

প্রোফাইল ২: দ্য ডেইলি ড্রাইভার (The Daily Driver) বা দ্য ভল্ট (The Vault) এটি হলো সেই প্রোফাইল যেখানে আপনি আপনার বেশিরভাগ সময় কাটাবেন। এটি আপনার সিকিউর কমিউনিকেশন হাব (secure communications hub) এবং প্রাইভেট ওয়ার্কস্পেস (private workspace)।

- **দ্য কন্টেন্টস (The Contents):** এই প্রোফাইলটিতে শুধুমাত্র হাইলি ট্রাস্টেড (highly trusted), প্রাইভেসি-রেস্পেক্টিং (privacy-respecting) এবং ওপেন-সোর্স (open-source) অ্যাপ্লিকেশন থাকা উচিত। এখানেই আপনি সিকিউর মেসেজিংয়ের জন্য **Signal**, এনক্রিপ্টেড ইমেইলের জন্য **ProtonMail** বা **Tuta**, আপনার প্রাইমারি পাসওয়ার্ড ম্যানেজার এবং সিকিউর ওয়েব রিসার্চের জন্য বিল্ট-ইন **Vanadium** ব্রাউজার ইনস্টল করবেন।
- **দ্য রুল (The Rule):** আপনি কোনো অবস্থাতেই এই প্রোফাইলে Sandboxed Google Play ইনস্টল করবেন না। আপনি সোশ্যাল মিডিয়াও ইনস্টল করবেন না। এই কম্পার্টমেন্টটি কঠোরভাবে সিকিউর এবং ভেরিফায়েবল অপারেশনগুলোর (verifiable operations) জন্য সংরক্ষিত।

প্রোফাইল ৩: দ্য ফাইন্যান্সিয়াল ও কর্পোরেট সেক্টর (The Financial & Corporate Sector) বা দ্য প্লে স্যান্ডবক্স (The Play Sandbox) আধুনিক জীবনে হোস্টাইল (hostile) বা ক্লোজড-সোর্স (closed-source) অ্যাপ্লিকেশনগুলোর সাথে ইন্টার‍্যাকশন (interaction) করা অনিবার্য। আপনার হয়তো মোবাইল ব্যাংকিং, রাইড-শেয়ারিং অ্যাপ (যেমন—Uber বা Lyft) এবং এমন কর্পোরেট কমিউনিকেশন টুলসের (যেমন—Microsoft Teams বা Slack) প্রয়োজন হতে পারে, যা আপনার এমপ্লয়ারের (employer) দ্বারা বাধ্যতামূলক। এই অ্যাপ্লিকেশনগুলো অ্যাগ্রেসিভলি (aggressively) টেলিমেট্রি (telemetry) ট্র্যাক করে এবং এগুলো কাজ করার জন্য প্রায় এক্সক্লুসিভলি (exclusively) Google Play Services-এর প্রয়োজন হয়।

- **দ্য কন্টেন্টস (The Contents):** এই প্রোফাইলটিকে কোয়ারেন্টাইন জোন (quarantine zone) হিসেবে ডিজাইন করা হয়েছে। এখানেই আপনি **Sandboxed Google Play** কম্প্যাটিবিলিটি লেয়ারটি (compatibility layer) ইনস্টল করবেন (যেমনটি সেকশন ২.১-এ আলোচনা করা হয়েছে)। স্যান্ডবক্সটি এস্টাবলিশ (establish) হওয়ার পর,

আপনি আপনার ব্যাংকিং অ্যাপ, রাইড-শেয়ারিং অ্যাপ এবং প্রোপ্রাইটারি কর্পোরেট সফটওয়্যার (proprietary corporate software) ইনস্টল করবেন।

- **দ্য রুল (The Rule):** এই অ্যাপগুলোকে এখানে আইসোলেট (isolate) করার মাধ্যমে, আপনি সেগুলোকে Sandboxed Google Play ব্যবহার করে পারফেক্টলি ফাংশন করার অনুমতি দিচ্ছেন, কিন্তু একইসাথে প্রোফাইল ২-এ থাকা আপনার সিকিউর কমিউনিকেশনগুলো থেকে সেগুলোকে সম্পূর্ণ ব্লাইন্ড (blind) করে রাখছেন। আপনার ব্যাংকিং অ্যাপটি স্ক্যান করে দেখতে পারবে না যে আপনার ডিভাইসে Signal ইনস্টল করা আছে, এবং আপনার কর্পোরেট স্পাইওয়্যার (corporate spyware) আপনার পার্সোনাল ফটোগুলোতে অ্যাক্সেস করতে পারবে না।

প্রোফাইল ৪: দ্য ট্রাভেল বা বার্নার প্রোফাইল (The Travel / Burner Profile) বা দ্য এয়ারলক (The Airlock)
ইন্টারন্যাশনাল বর্ডার ক্রসিংগুলো (International border crossings) হলো লিগ্যালি কমপ্লেক্স (legally complex) এবং হোস্টাইল এনভায়রনমেন্ট (hostile environments)। অনেক জুরিসডিকশনে (jurisdictions), বর্ডার এজেন্টদের কাছে আপনার ডিভাইসটি আনলক (unlock) করতে বাধ্য করার মতো আইনি ক্ষমতা রয়েছে। আপনি যদি অস্বীকার করেন, তবে তারা হার্ডওয়্যারটি সিজ (seize) করতে পারে।

- **দ্য কন্টেন্টস (The Contents):** বার্নার প্রোফাইলটি (Burner profile) হলো একটি স্পার্স (sparse), "ডামি" এনভায়রনমেন্ট ("dummy" environment)। এতে কয়েকটি বিনাইন (benign), সাসপিশনহীন (unsuspicious) অ্যাপ্লিকেশন থাকা উচিত—হতে পারে একটি ওয়েদার (weather) অ্যাপ, আপনার ডেস্টিনেশনের একটি জেনেরিক অফলাইন ম্যাপ এবং একটি ক্লিন ওয়েব ব্রাউজার। এটির নিজস্ব একটি ইউনিক ও সিম্পল পিন (PIN) থাকা উচিত।
- **দ্য রুল (The Rule):** কোনো বর্ডার চেকপয়েন্টের কাছে পৌঁছানোর আগে বা কোনো হাই-রিস্ক ফিজিক্যাল এনভায়রনমেন্টে প্রবেশ করার আগে, আপনি সমস্ত সিকিউর প্রোফাইল (প্রোফাইল ১, ২ এবং ৩) থেকে লগ আউট করবেন, যা মেমোরি থেকে সেগুলোর এনক্রিপশন কি'গুলোকে সম্পূর্ণভাবে পার্জ (purge) করে দেবে। এরপর আপনি বার্নার প্রোফাইলটিতে লগ-ইন করবেন। যদি কোনো হোস্টাইল অ্যাক্টর (hostile actor) বা বর্ডার এজেন্ট জোড়পূর্বক আপনার পিন (PIN) দাবি করে, তবে আপনি বার্নার প্রোফাইলের পিনটি প্রদান করবেন। তারা ডিভাইসটি আনলক করবে, একটি বিনাইন ও প্রায় খালি এনভায়রনমেন্ট দেখতে পাবে এবং আপনার সিকিউর কমিউনিকেশন বা ফাইন্যান্সিয়াল ডেটার একেবারে জিরো এভিডেন্স (zero evidence) খুঁজে পাবে।

দ্য ফ্রিকশন অব সিকিউরিটি (The Friction of Security)

মাল্টিপল প্রোফাইল (multiple profiles) রান করানোর ধারণাটি অনেক ইউজার প্রাথমিকভাবে মেনে নিতে পারেন না। তারা এই "ফ্রিকশন" (friction) বা বাধা নিয়ে অভিযোগ করেন। শুধুমাত্র ব্যাংক ব্যালেন্স চেক করতে বা উবার (Uber) ডাকতে ডেইলি ড্রাইভার (Daily Driver) প্রোফাইল থেকে ফাইন্যান্সিয়াল (Financial) প্রোফাইলে সুইচ করাটা তাদের কাছে বিরক্তিকর মনে হয়।

আপনাকে অবশ্যই ফ্রিকশন সম্পর্কে আপনার আন্ডারস্ট্যান্ডিংটি (understanding) রিপ্রোগ্রাম (reprogram) করতে হবে।

সাইবার সিকিউরিটিতে ফ্রিকশন হলো এমন একটি ডিফেন্স মেকানিজমের (defense mechanism) ফিজিক্যাল ম্যানিফেস্টেশন (physical manifestation), যা ঠিক সেভাবেই কাজ করছে যেভাবে এটিকে ডিজাইন করা হয়েছে। একটি ভারী স্টিলের ভল্ট ডোর (vault door) খোলা কঠিন; এতে ফ্রিকশন রয়েছে। আর ঠিক এ কারণেই এটি সিকিউর (secure)।

যখন আপনি স্ক্রিনে ট্যাপ করতে, নোটিফিকেশন শেড (notification shade) নিচে নামাতে এবং কোনো কর্পোরেট অ্যাপ্লিকেশন ওপেন করার জন্য এক্সপ্লিসিটলি (explicitly) প্রোফাইল সুইচ করতে বাধ্য হন, তখন আপনি ফিজিক্যালি বাল্কহেডটিকে ম্যানিপুলেট (manipulate) করছেন। আপনি অ্যাক্টিভলি (actively) সিকিউর কম্পার্টমেন্ট থেকে বেরিয়ে আসছেন এবং সাময়িকভাবে কোয়ারেন্টাইন জোনে প্রবেশ করছেন।

ফ্রিকশনের এই সামান্য মুহূর্তটিই হলো সেই সঠিক গাণিতিক ব্যারিয়ার (mathematical barrier), যা আপনার রাইড-শেয়ারিং অ্যাপের একটি জিরো-ডে এক্সপ্লয়েটকে (zero-day exploit) আপনার ফাইল সিস্টেমের ভেতর দিয়ে নীরবে প্রবেশ করে আপনার এনক্রিপ্টেড ইমেইলগুলো চুরি করতে বাধা দেয়। কম্পার্টমেন্টালাইজেশনকে গ্রহণ করুন। বাল্কহেডগুলো তৈরি করুন। নিশ্চিত করুন যে জাহাজটি যেন ডুবতে না পারে।

২.৭ ফাইন্যান্সিয়াল সিকিউরিটি (Financial Security) ও প্লে ইন্টিগ্রিটি (Play Integrity)

অ্যাডভান্সড অপারেশনাল সিকিউরিটির (advanced operational security) ক্ষেত্রে, গ্রেটেস্ট পয়েন্ট অব ফ্রিকশন (greatest point of friction) খুব কম ক্ষেত্রেই স্বয়ং অপারেটিং সিস্টেম হয়ে থাকে; বরং এটি হলো একটি হার্ডেনড (hardened) ওএস এবং পুওরলি কোডেড (poorly coded) কমার্শিয়াল অ্যাপ্লিকেশনের মধ্যে ঘটা ইন্টার‍্যাকশন (interaction)। ফাইন্যান্সিয়াল সেক্টরের (financial sector) চেয়ে বেশি এটি আর কোথাও এতটা স্পষ্ট নয়।

আপনি যদি সেকশন ২.৬-এর আর্কিটেকচারাল ব্লুপ্রিন্ট (architectural blueprint) অনুসরণ করে থাকেন, তবে আপনি স্যান্ডবক্সড গুগল প্লে (Sandboxed Google Play) কম্প্যাটিবিলিটি লেয়ার (compatibility layer) ব্যবহার করে একটি ডেডিকেটেড "Banking" প্রোফাইল তৈরি করেছেন। আপনি আপনার মোবাইল ব্যাংকিং অ্যাপ্লিকেশনটি ইনস্টল করে লঞ্চ (launch) করলেন এবং তাৎক্ষণিকভাবেই একটি উজ্জ্বল লাল রঙের ওয়ার্নিং স্ক্রিন দেখতে পেলেন:

"Error: Your device appears to be rooted, jailbroken, or modified. For your security, this application cannot run."

একজন নতুন GrapheneOS অপারেটরের কাছে এই মেসেজটি চরমভাবে বিভ্রান্তিকর। আপনি সবেমাত্র ঘণ্টার পর ঘণ্টা ব্যয় করে এই গ্রহের সবচেয়ে গাণিতিকভাবে সুরক্ষিত, অ্যাটাক-সারফেস-রিডিউসড (attack-surface-reduced) অপারেটিং সিস্টেমটি ইনস্টল করেছেন। আপনার বুটলোডার (bootloader) ক্রিপ্টোগ্রাফিক্যালি লকড (cryptographically locked) এবং আপনার ফাইল সিস্টেমটি হেভিলি এনক্রিপ্টেড (heavily encrypted)। আপনার ডিভাইসটি অবজেক্টিভলি (objectively) একটি স্ট্যান্ডার্ড কমার্শিয়াল ফোনের তুলনায় হাজার গুণ বেশি সুরক্ষিত।

তাহলে আপনার ব্যাংক কেন আপনার বিরুদ্ধে একটি কম্প্রোমাইজড (compromised), "রুটেড" ("rooted") ডিভাইস চালানোর অভিযোগ করছে?

এই আয়রনি (irony) বুঝতে হলে, আমাদের গুগলের রিমোট অ্যাটেস্টেশন সার্ভিসের (remote attestation services) মেকানিক্সের (mechanics) গভীরে প্রবেশ করতে হবে।

দ্য প্লে ইন্টিগ্রিটি এপিআই (The Play Integrity API - পূর্বে SafetyNet)

যখন একটি ব্যাংকিং অ্যাপ্লিকেশন লঞ্চ হয়, এটি নিজের সিকিউরিটি স্ট্যাটাস রিপোর্ট করার জন্য অপারেটিং সিস্টেমের ওপর আস্থা রাখে না। এর পরিবর্তে, ব্যাংকিং অ্যাপটি **প্লে ইন্টিগ্রিটি এপিআই** (**Play Integrity API**) (ঐতিহাসিকভাবে SafetyNet Attestation API নামে পরিচিত) নামক একটি ফ্রেমওয়ার্ক ব্যবহার করে রিমোট গুগল সার্ভারকে পিং (ping) করে।

প্লে ইন্টিগ্রিটি এপিআই ডিভাইসের ফিজিক্যাল সিলিকনকে (physical silicon - সিকিউর এলিমেন্ট বা secure element) হার্ডওয়্যার এবং সফটওয়্যার স্টেটের (state) একটি ক্রিপ্টোগ্রাফিক প্রুফ (cryptographic proof) জেনারেট করার নির্দেশ দেয় এবং এটি গুগলের কাছে ফেরত পাঠায়। গুগল এই প্রুফটি অ্যানালাইজ করে এবং ব্যাংকিং অ্যাপের কাছে একটি ভার্ডিক্ট (verdict) বা রায় প্রদান করে।

ভার্ডিক্টটি টিয়ারড ইন্টিগ্রিটি লেভেলে (tiered integrity levels) বিভক্ত করা হয়: ১. **MEETS_BASIC_INTEGRITY (পূর্বে `basicIntegrity`):** ডিভাইসটি রুটেড নয়, Android ইমুলেটর (emulator) ব্যবহার করা হচ্ছে না এবং কোনো জেনেরিক হুকিং ফ্রেমওয়ার্ক (generic hooking frameworks) (যেমন- Magisk বা Xposed) ডিটেক্ট করা যায়নি। ২. **MEETS_DEVICE_INTEGRITY (পূর্বে `ctsProfileMatch`):** ডিভাইসটি একটি অফিশিয়াল, গুগল-সার্টিফায়েড অপারেটিং সিস্টেম রান করছে যা হুবহু ফ্যাক্টরি কনফিগারেশনের (factory configuration) সাথে ম্যাচ করে এবং বুটলোডারটি অরিজিনাল ম্যানুফ্যাকচারারের ক্রিপ্টোগ্রাফিক কি দিয়ে লক করা রয়েছে। ৩. **MEETS_STRONG_INTEGRITY:** এটি হার্ডওয়্যার-ব্যাকড প্রুফের (hardware-backed proof) সর্বোচ্চ লেভেল, যা নিশ্চিত করে যে ফ্যাক্টরি ওএস (factory OS) সম্পূর্ণ আনঅল্টারড (unaltered) বা অপরিবর্তিত রয়েছে।

যখন GrapheneOS একটি প্লে ইন্টিগ্রি চেক এক্সিকিউট (execute) করে, এটি সফলভাবে `MEETS_BASIC_INTEGRITY` পাস করে। ডিভাইসটি স্ট্রাকচারালি সাউন্ড (structurally sound), বুটলোডারটি লকড এবং এতে কোনো রুট অ্যাক্সেস উপস্থিত নেই।

যাইহোক, গাণিতিকভাবে GrapheneOS `MEETS_DEVICE_INTEGRITY`-তে *ফেইল* (fails) করে।

কেন? কারণ GrapheneOS মিথ্যা বলতে অস্বীকৃতি জানায়। যখন প্লে ইন্টিগ্রিটি এপিআই সিকিউর এলিমেন্টকে কোয়েরি (queries) করে, তখন হার্ডওয়্যারটি বিশ্বস্ততার সাথে রিপোর্ট করে যে বর্তমানে ফ্যাক্টরি-ডিফল্ট গুগল কি-এর পরিবর্তে একটি কাস্টম Android ভেরিফায়েড বুট (Android Verified Boot বা AVB) কি লোড করা রয়েছে। ডিভাইসটি কম্প্রোমাইজড নয়, কিন্তু গুগলের কমার্শিয়াল ইকোসিস্টেমের (commercial ecosystem) দৃষ্টিতে এটি "আনসার্টিফায়েড" ("uncertified")।

যেহেতু অনেক ব্যাংকিং অ্যাপ্লিকেশন লেজিলি প্রোগ্রামড (lazily programmed), সেগুলো একটি হাইলি সিকিউর কাস্টম ওএস এবং একটি বিপজ্জনক রুটেড ওএস-এর মধ্যে পার্থক্য করতে পারে না। অ্যাপটি যদি দেখে যে `MEETS_DEVICE_INTEGRITY` ফেইল করেছে, তবে এটি সিম্পলি একটি জেনেরিক "Root Detected" এরর থ্রো (throw) করে এবং ক্র্যাশ (crashes) করে যায়।

স্পুফ করতে অস্বীকৃতি (The Refusal to Spoof)

এই ফ্রিকশনটির কারণে অনেক কাস্টম Android রম (যেমন- LineageOS) **ডিভাইস স্পুফিং** (**Device Spoofing**) নামক একটি বিপজ্জনক প্র্যাকটিসে যুক্ত হয়েছে।

ব্যাংকিং অ্যাপগুলোকে ট্রিক (trick) করে কাজ করানোর জন্য, তুলনামূলক দুর্বল এই অপারেটিং সিস্টেমগুলো ইনটেনশনালি (intentionally) এমন কোড ইনজেক্ট (inject) করে যা প্লে ইন্টিগ্রিটি এপিআই রিকোয়েস্টকে ইন্টারসেপ্ট (intercept) করে। ইনজেক্ট করা কোডটি গুগলের কাছে মিথ্যা বলে, হার্ডওয়্যার আইডেন্টিফায়ার (hardware identifiers) এবং ক্রিপ্টোগ্রাফিক সিগনেচার (cryptographic signatures) স্পুফ (spoofing) করে যাতে ডিভাইসটিকে হুবহু একটি ফ্যাক্টরি-লকড (factory-locked) Google Pixel-এর মতো মনে হয়।

GrapheneOS অ্যাগ্রেসিভলি (aggressively) এবং ক্যাটেগরিক্যালি (categorically) ডিভাইস স্পুফিং ইমপ্লিমেন্ট (implement) করতে অস্বীকৃতি জানায়।

এলিট সাইবারসিকিউরিটিতে, ক্রিপ্টোগ্রাফিক ইন্টিগ্রিটি (cryptographic integrity) হলো অ্যাবসলিউট (absolute)। অপারেটিং সিস্টেমটিকে যদি এমনভাবে ডিজাইন করা হয় যা নির্বিঘ্নে হার্ডওয়্যার আইডেন্টিফায়ার স্পুফ করতে পারে এবং গুগলের কাছে তার ভেরিফায়েড বুট স্টেট (verified boot state) সম্পর্কে মিথ্যা বলতে পারে, তবে এটিকে ম্যানিপুলেট (manipulate) করে *আপনার* কাছেও মিথ্যা বলা সম্ভব। একটি ব্যাংকিং অ্যাপকে সন্তুষ্ট করার জন্য হার্ডওয়্যার-ব্যাকড সিকিউরিটি মডেল (hardware-backed security models) বাইপাস করা প্ল্যাটফর্মটির প্রতি ফান্ডামেন্টাল ট্রাস্টকে (fundamental trust) সম্পূর্ণভাবে ধ্বংস করে দেয়। GrapheneOS ভেরিফায়েড ট্রুথ (verified truth)-এর ফিলোসফির ওপর নির্মিত, কনভেনিয়েন্ট ডিসেপশনের (convenient deception) ওপর নয়।

ফলস্বরূপ, হাইলি অ্যাগ্রেসিভ ফাইন্যান্সিয়াল অ্যাপ্লিকেশনের একটি ছোট সাবসেট (subset) (যেমন- Google Wallet-এর tap-to-pay ফাংশনালিটি বা কিছু রিজিওনাল ব্যাংকিং অ্যাপ) সিম্পলি GrapheneOS-এ রান করতে অস্বীকার করবে। একজন অপারেটর হিসেবে, আপনাকে অবশ্যই এই বাস্তবতাটি মেনে নিতে হবে। আপনাকে হয়তো এই নির্দিষ্ট ব্যাংকগুলোর নেটিভ অ্যাপ্লিকেশনের (native applications) পরিবর্তে Vanadium ওয়েব ব্রাউজারের মাধ্যমে অ্যাক্সেস করতে হতে পারে।

বাগি কোড বনাম এক্সপ্লয়েটস (Buggy Code vs. Exploits): `hardened_malloc`

কখনো কখনো, কোনো ব্যাংকিং অ্যাপ্লিকেশন লঞ্চ করার সাথে সাথেই ক্র্যাশ করতে পারে, তবে এর সাথে প্লে ইন্টিগ্রিটি এপিআই-এর কোনো সম্পর্ক নেই।

GrapheneOS `hardened_malloc` নামে পরিচিত একটি প্রোপ্রাইটারি (proprietary), হাইলি অ্যাগ্রেসিভ মেমোরি অ্যালোকেটর (memory allocator) ব্যবহার করে। পূর্বে আলোচনা করা হয়েছে যে, এই অ্যালোকেটরটি তাৎক্ষণিকভাবে মেমোরি করাপশন ভালনারেবিলিটি (memory corruption vulnerabilities)—যেমন বাফার ওভারফ্লো (buffer overflows) বা ইউজ-আফটার-ফ্রি (use-after-free) বাগগুলো—ডিটেক্ট করার জন্য ডিজাইন করা হয়েছে এবং একটি জিরো-ডে এক্সপ্লয়েট কোনো ম্যালিশিয়াস পেলোড (malicious payload) এক্সিকিউট করার আগেই অ্যাপ্লিকেশনটিকে ফোর্সফুলি টার্মিনেট (forcefully terminate) করে।

কমার্শিয়াল সফটওয়্যার ইন্ডাস্ট্রি (commercial software industry) পুওরলি রিটেন কোডে (poorly written code) ভরপুর। অনেক ব্যাংকিং অ্যাপ এবং কর্পোরেট টুলে (corporate tools) ল্যাটেন্ট (latent) ও অ্যাকসিডেন্টাল মেমোরি করাপশন বাগ (accidental memory corruption bugs) রয়েছে। একটি স্ট্যান্ডার্ড Android ডিভাইসে, এই বাগগুলো ব্যাকগ্রাউন্ডে নীরবে অবস্থান করে, যা ডিভাইসটিকে এক্সপ্লয়েটেশনের (exploitation) জন্য হাইলি ভালনারেবল (highly vulnerable) করে তোলে। অন্যদিকে GrapheneOS-এ, `hardened_malloc` এই বাগগুলোকে তাৎক্ষণিকভাবে ডিটেক্ট করে এবং সিস্টেমকে রক্ষা করার জন্য অ্যাপটিকে রুথলেসলি (ruthlessly) ক্র্যাশ করে দেয়।

আপনার যদি এমন কোনো অ্যাপ্লিকেশন থাকে যা ওপেন করার সাথে সাথেই ক্র্যাশ করে যায়, তবে অত্যন্ত সম্ভাবনা রয়েছে যে GrapheneOS ডেভেলপারের কোডে একটি মেমোরি করাপশন বাগ ধরে ফেলেছে।

এক্সপ্লয়েট প্রোটেকশন কম্প্যাটিবিলিটি মোড (Exploit Protection Compatibility Mode)

যদি `hardened_malloc`-এর কারণে কোনো ক্রিটিক্যাল অ্যাপ্লিকেশন ক্র্যাশ করে এবং আপনাকে দৈনন্দিন অপারেশনের জন্য এটি অবশ্যই ব্যবহার করতে হয়, তবে GrapheneOS একটি ট্যাকটিকাল বাইপাস টগল (tactical bypass toggle) প্রদান করে।

১. **Settings > Apps**-এ নেভিগেট (Navigate) করুন এবং ক্র্যাশ করা অ্যাপ্লিকেশনটি সিলেক্ট করুন। ২. **Exploit protection compatibility mode** টগলটি খুঁজে বের করুন। ৩. সুইচটি টগল করে **ON** করুন।

আপনি যখন এই মোডটি এনাবল (enable) করেন, তখন GrapheneOS মেমোরি অ্যালোকেশন রুলগুলোকে (memory allocation rules) *স্পেসিফিকভাবে কেবল সেই একটি অ্যাপের জন্য* ডাউনগ্রেড (downgrades) করে। এটি অ্যাপটিকে `hardened_malloc` থেকে স্ট্যান্ডার্ড Android মেমোরি অ্যালোকেটরে (Scudo) ব্যাক করায় এবং অ্যাডভান্সড হার্ডওয়্যার মেমোরি ট্যাগিং (advanced hardware memory tagging) ডিজেবল (disable) করে দেয়।

> ***OpSec Warning:** Exploit Protection Compatibility Mode এনাবল করা একটি ক্যালকুলেটেড রিস্ক (calculated risk)। অ্যাপ্লিকেশনটির ক্র্যাশ করা বন্ধ হবে এবং স্বাভাবিকভাবে কাজ করতে শুরু করবে, কিন্তু আপনি ইনটেনশনালি (intentionally) সেই নির্দিষ্ট কম্পার্টমেন্টটির জন্য শিল্ড (shields) কমিয়ে দিয়েছেন। সেই অ্যাপ্লিকেশনটি এখন জিরো-ডে মেমোরি এক্সপ্লয়েটের (zero-day memory exploits) জন্য অত্যন্ত ভালনারেবল।*

আপনার অত্যন্ত প্রয়োজনীয় এবং হাইলি ট্রাস্টেড অ্যাপের (highly trusted apps) জন্যই কেবল এই টগলটি ডিপ্লয় (deploy) করা উচিত এবং আপনাকে অবশ্যই সেগুলোকে একটি কোয়ারেন্টাইনড ইউজার প্রোফাইলের (quarantined User Profile) মধ্যে আইসোলেট করতে হবে (যেমনটি সেকশন ২.৬-এ বিস্তারিত আলোচনা করা হয়েছে)।

একটি হার্ডেনড ডিভাইসে ফাইন্যান্সিয়াল সেক্টর নেভিগেট করার জন্য ধৈর্যের প্রয়োজন। আপনি ফ্রিকশন, ভুল "root" অ্যাকুইজেশন (accusations) এবং ক্রাশিং কোডের সম্মুখীন হবেন। কিন্তু ডকট্রিনটি (doctrine) মনে রাখবেন: কনভেনিয়েন্স (convenience) হলো একটি ভালনারেবিলিটি। আপনার অভিজ্ঞতায় থাকা এই ফ্রিকশনগুলোই হলো সেই গাণিতিক প্রমাণ, যা নির্দেশ করে যে ফোর্ট্রেসটি (fortress) টিকে আছে।

২.৮ ডিফেন্ডিং অ্যাগেইনস্ট ফিশিং (Defending Against Phishing): অ্যাপ লিংক ভেরিফিকেশন (App Link Verification)

সাইবারসিকিউরিটির ইতিহাসে ফিশিং (Phishing) হলো সবচেয়ে পুরোনো এবং সবচেয়ে রিলায়েবল (reliable) অ্যাটাক ভেক্টর (attack vector)। এর প্রিমাইসটি (premise) খুবই সাধারণ: আপনার ডিভাইসের এনক্রিপশন (encryption) বাইপাস করার জন্য মিলিয়ন মিলিয়ন ডলার খরচ করে একটি জিরো-ডে এক্সপ্লয়েট (zero-day exploit) ডেভেলপ (develop) করার চেয়ে, অ্যাডভারসারি (adversary) খুব সহজেই আপনার কাছে আপনার পাসওয়ার্ডটি চেয়ে নেয়।

ঐতিহাসিকভাবে, ফিশিং মূলত ইমেইলের মাধ্যমে সংঘটিত হতো। আপনি আপনার ব্যাংকের নকল করা একটি ইমেইল রিসিভ (receive) করতেন, সেখানে দেওয়া লিংকটিতে ক্লিক করতেন এবং আপনাকে একটি ফ্রডুলেন্ট ওয়েবসাইটে (fraudulent website) ডিরেক্ট (direct) করা হতো। যাইহোক, মডার্ন মোবাইল অপারেটিং সিস্টেমগুলোতে, ফিশিং থ্রেট মডেলটি (phishing threat model) বিবর্তিত হয়ে একটি অত্যন্ত সফিস্টিকেটেড (sophisticated), লোকাল অ্যাটাকে (local attack) পরিণত হয়েছে যা অপারেটিং সিস্টেমের নিজস্ব কোর আর্কিটেকচারকেই (core architecture) কাজে লাগায়।

GrapheneOS কীভাবে এই থ্রেটটিকে নিউট্রালাইজ (neutralizes) করে তা বোঝার জন্য, আমাদের প্রথমেই Android-এর ফান্ডামেন্টাল কমিউনিকেশন প্রোটোকল (fundamental communication protocol) পরীক্ষা করতে হবে, যা হলো: **ইন্টেন্টস** (**Intents**)।

Android ইন্টেন্টস-এর মেকানিক্স (The Mechanics of Android Intents)

অপারেটিং সিস্টেমগুলো কম্পার্টমেন্টালাইজেশনের (compartmentalization) ওপর ভিত্তি করে নির্মিত হয়। যখন আপনি একটি এনক্রিপ্টেড মেসেজিং অ্যাপ ব্যবহার করছেন, তখন এটি নিজের আইসোলেটেড মেমোরি স্যান্ডবক্সের (isolated memory sandbox) ভেতরে রান করে। যখন কেউ আপনাকে একটি ওয়েব লিংক (যেমন, `https://bank.com/login`) পাঠায় এবং আপনি সেটিতে ট্যাপ (tap) করেন, তখন মেসেজিং অ্যাপটি চাইলেই আপনার ওয়েব ব্রাউজারের কন্ট্রোল বা নিয়ন্ত্রণ নিতে পারে না।

এর পরিবর্তে, এটি ইন্টার-প্রসেস কমিউনিকেশন (Inter-Process Communication বা IPC) মেকানিজম ব্যবহার করে, যা একটি **ইন্টেন্ট** (**Intent**) হিসেবে পরিচিত।

যখন আপনি লিংকটিতে ট্যাপ করেন, তখন মেসেজিং অ্যাপটি পুরো অপারেটিং সিস্টেমের কাছে একটি ইন্টেন্ট ব্রডকাস্ট (broadcast) করে, যা মূলত চিৎকার করে বলে: *"ইউজার এই URL-টিতে ট্যাপ করেছে। এই ডিভাইসের কোনো অ্যাপ্লিকেশন কি জানে এটিকে কীভাবে হ্যান্ডেল (handle) করতে হয়?"*

আপনার কাছে যদি অফিশিয়াল `bank.com` অ্যাপ্লিকেশনটি ইনস্টল করা থাকে, তবে সেই অ্যাপটি তার ডিজিটাল হাত তুলবে এবং ডিক্লেয়ার (declare) করবে, *"আমি এই ডোমেইনটি (domain) হ্যান্ডেল করার জন্য রেজিস্টার্ড (registered)।"* এরপর অপারেটিং সিস্টেমটি অত্যন্ত নির্বিঘ্নে URL-টিকে ব্যাংকিং অ্যাপের হাতে তুলে দেয় এবং অ্যাপটি সরাসরি লগইন স্ক্রিনে (login screen) ওপেন হয়। এটি অবিশ্বাস্যভাবে স্মুথ (smooth) এবং সুবিধাজনক একটি ইউজার এক্সপেরিয়েন্স (user experience)।

দ্য হাইজ্যাক অ্যাটাক ভেক্টর (The Hijack Attack Vector)

দুর্ভাগ্যবশত, স্ট্যান্ডার্ড কমার্শিয়াল Android ইমপ্লিসিট ট্রাস্টের (implicit trust) একটি মডেলে কাজ করে। এটি একটি ডেভাস্টেটিং ভালনারেবিলিটির (devastating vulnerability) জন্ম দেয়, যা **ইন্টেন্ট হাইজ্যাকিং** (**Intent Hijacking**) নামে পরিচিত।

একটি স্ট্যান্ডার্ড ডিভাইসের যেকোনো অ্যাপ্লিকেশন যেকোনো ডোমেইনের জন্য একটি ইন্টেন্ট ফিল্টার (Intent Filter) রেজিস্টার করতে পারে।

কল্পনা করুন, আপনি অসাবধানতাবশত একটি পুওরলি কোডেড, থার্ড-পার্টি PDF রিডার বা একটি স্কেচি (sketchy) মোবাইল গেম ইনস্টল করেছেন। সেই ম্যালিশিয়াস (malicious) অ্যাপ্লিকেশনটি গোপনে অপারেটিং সিস্টেমের কাছে একটি ইন্টেন্ট ফিল্টার রেজিস্টার করে দাবি করতে পারে যে এটি `https://bank.com`-এর অফিশিয়াল হ্যান্ডলার।

পরবর্তীতে যখন আপনি আপনার ব্যাংকের কোনো লেজিটিমেট (legitimate) ইমেইল বা টেক্সট মেসেজ লিংকসহ রিসিভ করবেন এবং সেটিতে ট্যাপ করবেন, তখন অপারেটিং সিস্টেমটি একটি ইন্টেন্ট ব্রডকাস্ট করবে। আর সেই ম্যালিশিয়াস PDF রিডারটি রিকোয়েস্টটিকে ইন্টারসেপ্ট (intercept) করবে এবং তাৎক্ষণিকভাবে আপনার ব্যাংকের লগইন পেজের একটি ফুল-স্ক্রিন, পিক্সেল-পারফেক্ট (pixel-perfect) ক্লোন (clone) লঞ্চ করবে।

আপনি মনে করবেন যে আপনি আপনার ব্যাংকিং অ্যাপ বা আপনার সিকিউর ওয়েব ব্রাউজারের সাথে ইন্টার‍্যাক্ট করছেন। আপনি আপনার ইউজারনেম, পাসওয়ার্ড এবং আপনার টু-ফ্যাক্টর অথেনটিকেশন (two-factor authentication) কোড টাইপ করবেন। ম্যালিশিয়াস অ্যাপটি তৎক্ষণাৎ সেই ক্রেডেনশিয়ালগুলোকে (credentials) একটি কমান্ড-অ্যান্ড-কন্ট্রোল সার্ভারে (command-and-control server) এক্সফিলট্রেট (exfiltrates) করবে এবং সন্দেহ এড়ানোর জন্য একটি ভুয়া "Network Error" প্রদর্শন করবে।

অ্যাডভারসারির কখনোই ব্যাংক হ্যাক করার প্রয়োজন হয়নি এবং তাদের আপনার এনক্রিপশন ভাঙারও কোনো প্রয়োজন ছিল না। তারা সিম্পলি URL হ্যান্ডঅফটিকে (handoff) ইন্টারসেপ্ট করেছে।

GrapheneOS ডিফেন্স: স্ট্রিক্ট ভেরিফিকেশন (Strict Verification)

GrapheneOS অ্যাগ্রেসিভভাবে **অ্যাপ লিংক ভেরিফিকেশন** (**App Link Verification**) এনফোর্স (enforcement) করার মাধ্যমে এই সম্পূর্ণ অ্যাটাক ভেক্টরটিকে নির্মূল করে।

GrapheneOS ইমপ্লিসিট ট্রাস্টের ওপর কাজ করতে অস্বীকৃতি জানায়। যখন কোনো অ্যাপ্লিকেশন ইনস্টল করা হয় এবং সেটি দাবি করে যে এটি `bank.com`-এর জন্য URL হ্যান্ডেল করতে পারে, তখন GrapheneOS শুধু অ্যাপ্লিকেশনটির কথায় বিশ্বাস করে না। এটি ওনারশিপের (ownership) ক্রিপ্টোগ্রাফিক প্রুফ (cryptographic proof) দাবি করে।

এই ভেরিফিকেশনটি (verification) ডিজিটাল অ্যাসেট লিংকস (Digital Asset Links) নামক একটি প্রোটোকলের (protocol) ওপর নির্ভর করে। আপনি যখন অফিশিয়াল ব্যাংকিং অ্যাপ্লিকেশনটি ইনস্টল করেন, তখন ওএস (OS) নীরবে একটি সিকিউর TLS কানেকশনের মাধ্যমে প্রকৃত `bank.com` ওয়েব সার্ভারে পৌঁছায়। এটি সার্ভারের রুট ডিরেক্টরিতে (root directory) একটি নির্দিষ্ট, ক্রিপ্টোগ্রাফিক্যালি সাইনড (cryptographically signed) মেটাডেটা ফাইল (`assetlinks.json`)-এর জন্য সার্চ করে।

এই ফাইলটিতে অফিশিয়াল Android অ্যাপ্লিকেশনটির এক্স্যাক্ট ডিজিটাল সিগনেচার (exact digital signature) থাকে। অপারেটিং সিস্টেম আপনার ডিভাইসে ইনস্টল করা অ্যাপের সিগনেচারের সাথে ব্যাংকের প্রকৃত ওয়েব সার্ভারে হোস্টেড (hosted) সিগনেচারটিকে ক্রস-রেফারেন্স (cross-references) করে।

যদি সিগনেচারগুলো ম্যাচ (match) করে, তবে ওনারশিপের গাণিতিক প্রুফ প্রতিষ্ঠিত হয়। এরপর অপারেটিং সিস্টেমটি অ্যাপ্লিকেশনটিকে নির্বিঘ্নে লিংকগুলো ওপেন করার অনুমতি দেবে।

দ্য ভ্যানাডিয়াম কোয়ারেন্টাইন (The Vanadium Quarantine)

কিন্তু যখন ম্যালিশিয়াস PDF রিডারটি দাবি করে যে এটি `bank.com` লিংক হ্যান্ডেল করতে পারে, তখন কী ঘটে?

GrapheneOS ঠিক একই ভেরিফিকেশন প্রোটোকল এক্সিকিউট করে। এটি ব্যাংকের ওয়েব সার্ভারে পৌঁছায় এবং `assetlinks.json` ফাইলটি চেক করে। এটি তাৎক্ষণিকভাবেই আবিষ্কার করে যে, ম্যালিশিয়াস PDF রিডারের ডিজিটাল সিগনেচারটি ব্যাংকের দ্বারা হোস্ট করা অফিশিয়াল সিগনেচারের সাথে ম্যাচ করছে না।

ক্রিপ্টোগ্রাফিক প্রুফটি ফেইল (fails) করে।

ঠিক এই মুহূর্তে, GrapheneOS একটি হার্ড ওভাররাইড (hard override) এক্সিকিউট করে। এটি ইন্টেন্ট ইন্টারসেপ্ট করার জন্য ম্যালিশিয়াস অ্যাপের রিকোয়েস্টটিকে সম্পূর্ণভাবে ইগনোর (ignores) করে। এর পরিবর্তে, ওএস (OS) URL-টিকে **Vanadium** ওয়েব ব্রাউজারের ভেতরে ওপেন হতে ফোর্স (forces) করে।

আনভেরিফায়েড লিংকটিকে (unverified link) Vanadium-এর ভেতরে ফোর্স করার মাধ্যমে, GrapheneOS এক্সিকিউশনটিকে (execution) একটি হার্ডেনড (hardened), আইসোলেটেড রেন্ডারার স্যান্ডবক্সের (isolated renderer sandbox) ভেতরে আটকে দেয় (যেমনটি সেকশন ১.৮-এ বিস্তারিত বলা হয়েছে)। যদি লিংকটি সত্যি সত্যিই একটি ফিশিং অ্যাটেম্পট (phishing attempt) হয়ে থাকে, তবে আপনি Vanadium অ্যাড্রেস বারে (address bar) আসল URL-টি দেখতে পাবেন, যা আপনাকে সহজেই ফ্রড (fraud) বা জালিয়াতিটি আইডেন্টিফাই (identify) করতে সাহায্য করবে। উপরন্তু, যেহেতু Vanadium কঠোরভাবে সাইট ডেটাকে আইসোলেট করে, সেহেতু এই ম্যালিশিয়াস লিংকটি আপনার আসল ব্যাংকের কোনো সিকিউর কুকিজ (secure cookies) বা সেশন টোকেনে (session tokens) অ্যাক্সেস পাবে না।

স্ট্রিক্ট অ্যাপ লিংক ভেরিফিকেশনের (strict App Link Verification) মাধ্যমে, GrapheneOS নিশ্চিত করে যে লোকাল ইন্টেন্ট হাইজ্যাকিং গাণিতিকভাবে অসম্ভব। আপনি কমিউনিকেশন প্রোটোকলগুলো ডিক্টেট (dictate) করেন এবং অপারেটিং সিস্টেম সেই সত্যকে এনফোর্স (enforces) করে।

২.৯ Physical Armor: Hardware Security Keys

এই অধ্যায় জুড়ে, আমরা নিয়মতান্ত্রিকভাবে আপনার অপারেটিং সিস্টেমের ইন্টারনাল আর্কিটেকচারকে শক্তিশালী করেছি। আমরা হোস্টাইল অ্যাপ্লিকেশনগুলোকে স্যান্ডবক্স (sandbox) করেছি, অনাকাঙ্ক্ষিত নেটওয়ার্ক সংযোগগুলো ছিন্ন করেছি এবং গাণিতিকভাবে আমাদের কমিউনিকেশন চ্যানেলগুলোকে যাচাই করেছি। তবে, বিশ্বের সবচেয়ে শক্তিশালী ডিজিটাল ফোর্ট্রেসটিও অরক্ষিত থেকে যাবে যদি অ্যাডভারসারি (adversary) কেবল আপনার কি (keys) চুরি করে নেয়।

কয়েক দশক ধরে, সাইবারসিকিউরিটি ইন্ডাস্ট্রি পাসওয়ার্ডের ওপর নির্ভরশীল ছিল। পাসওয়ার্ডগুলো চরমভাবে ব্যর্থ হয়েছে। এগুলো বারবার রিইউজ (reuse) করা হয়, বিশাল ডেটা ব্রিচগুলোতে (data breaches) লিক হয়ে যায় এবং সোশ্যাল ইঞ্জিনিয়ারিংয়ের (social engineering) মাধ্যমে খুব সহজেই এক্সট্র্যাক্ট করা যায়। এর প্রতিক্রিয়া হিসেবে, ইন্ডাস্ট্রি Two-Factor Authentication (2FA) এর দিকে অগ্রসর হয়েছে।

দুর্ভাগ্যবশত, সমস্ত 2FA সমানভাবে তৈরি হয় না। একটি এলিট সিকিউরিটি পোসচার (security posture) অর্জন করতে হলে, আমাদের অবশ্যই পুরনো অথেনটিকেশন প্রোটোকলগুলো (authentication protocols) পরিত্যাগ করতে হবে এবং একচেটিয়াভাবে ফিজিক্যাল ক্রিপ্টোগ্রাফিক আর্মার (physical cryptographic armor) অর্থাৎ **Hardware Security Keys**-এ ট্রান্সিশন করতে হবে।

SMS এবং TOTP এর Vulnerability

যখন কোনো অপারেটর প্রথমবার তাদের সিকিউরিটি আপগ্রেড করেন, তখন তারা সাধারণত পাসওয়ার্ড থেকে SMS-ভিত্তিক 2FA-তে ট্রান্সিশন করেন, অথবা সম্ভবত Aegis বা Google Authenticator এর মতো একটি Time-based One-Time Password (TOTP) অথেনটিকেটর অ্যাপ্লিকেশনে।

এই উভয় মেথডেই মারাত্মক আর্কিটেকচারাল ত্রুটি রয়েছে।

SMS (Text Message) 2FA হলো বর্তমানে ডেপ্লয়মেন্টে থাকা অথেনটিকেশনের সবচেয়ে দুর্বল রূপ। এটি **SIM Swapping** এর প্রতি অত্যন্ত সংবেদনশীল—এটি এমন একটি অ্যাটাক যেখানে একজন অ্যাডভারসারি কোনো স্বল্প বেতনের টেলিকম কর্মচারীকে ঘুষ দেয় বা সোশ্যাল ইঞ্জিনিয়ারিংয়ের মাধ্যমে আপনার ফোন নম্বরটি অ্যাটাকারের নিয়ন্ত্রিত একটি SIM কার্ডে পোর্ট করে নেয়। একবার নম্বরটি পোর্ট হয়ে গেলে, অ্যাডভারসারি আপনার সমস্ত 2FA টেক্সট মেসেজ রিসিভ করে এবং মুহূর্তের মধ্যে আপনার সিকিউরিটি বাইপাস করে ফেলে। SS7 নেটওয়ার্ক এক্সপ্লয়টেশনের (network exploitation) মাধ্যমে ইন্টারসেপ্ট (intercept) হওয়ার ক্ষেত্রেও SMS রাউটিং ভালনারেবল।

> ⚠ ***Warning:*** *কোনো হাই-ভ্যালু অ্যাকাউন্টে Two-Factor Authentication এর জন্য কখনোই SMS ব্যবহার করবেন না। SMS কোডগুলো প্লেইন টেক্সট (plain text) হিসেবে পাবলিক টেলিফোন নেটওয়ার্কের মধ্য দিয়ে যায় এবং SIM swap, SS7 exploitation বা Stingray ইন্টারসেপশনের মাধ্যমে এগুলো ইন্টারসেপ্ট করা সম্ভব। একটি Hardware Security Key (FIDO2) ব্যবহার করুন, অথবা অন্ততপক্ষে একটি TOTP অথেনটিকেটর অ্যাপ ব্যবহার করুন।*

TOTP (Authenticator Apps) SMS এর চেয়ে উল্লেখযোগ্যভাবে বেশি শক্তিশালী কারণ এগুলো টেলিকম নেটওয়ার্কের পরিবর্তে সরাসরি আপনার ডিভাইসে লোকালি সংরক্ষিত ক্রিপ্টোগ্রাফিক সিক্রেটের (cryptographic secrets) ওপর নির্ভর করে। তবে, TOTP অ্যাপগুলো এখনও **Man-in-the-Middle (MitM) Phishing** এর প্রতি চরমভাবে ভালনারেবল।

যদি একজন অত্যন্ত সোফিস্টিকেটেড অ্যাডভারসারি আপনাকে একটি পিক্সেল-পারফেক্ট, ফেক (fake) ব্যাংকিং ওয়েবসাইটে রিডাইরেক্ট করে, তাহলে আপনি সেখানে আপনার ইউজারনেম এবং পাসওয়ার্ড টাইপ করবেন। এরপর সেই ফেক ওয়েবসাইটটি আপনার ৬-ডিজিটের অথেনটিকেটর কোড চাইবে। আপনি আপনার অ্যাপটি ওপেন করবেন, কোডটি দেখবেন এবং সেই ফেক সাইটে টাইপ করবেন। সেই ভগ্নাংশ সেকেন্ডের মধ্যে, অ্যাডভারসারির সার্ভার তাৎক্ষণিকভাবে আপনার কোডটি আসল ব্যাংকের সাইটে ফরোয়ার্ড করবে, লগ ইন করবে, সেশন টোকেন (session token) চুরি করবে এবং আপনার অ্যাকাউন্টগুলো খালি করে দেবে।

যেহেতু *আপনি* নিজেই সেই মেকানিজম (mechanism) যা অ্যাপ থেকে কোডটি ওয়েবসাইটে ট্রান্সফার করছে, তাই হিউম্যান এররই (human error) ফিশিংকে (phishing) সম্ভব করে তোলে।

The Phishing-Proof Standard: FIDO2 এবং WebAuthn

MitM ফিশিং থ্রেটকে (threat) নিউট্রালাইজ করার জন্য, সাইবারসিকিউরিটি ইন্ডাস্ট্রি **FIDO2 / WebAuthn** স্ট্যান্ডার্ড ডেভেলপ করেছে। এই স্ট্যান্ডার্ডটি ডেডিকেটেড ফিজিক্যাল হার্ডওয়্যার টোকেনগুলোর (physical hardware tokens) ওপর নির্ভর করে—যার মধ্যে সবচেয়ে পরিচিত এবং হাইলি রিকমেন্ডেড হলো **YubiKey** (Yubico দ্বারা প্রস্তুতকৃত)।

একটি YubiKey হলো একটি ছোট, ফিজিক্যাল ক্রিপ্টোগ্রাফিক ভল্ট (cryptographic vault) যা আপনি আপনার ডিভাইসের USB-C পোর্টে প্লাগ করতে পারেন বা NFC (Near Field Communication) এর মাধ্যমে আপনার ফোনের পেছনে ট্যাপ করতে পারেন।

YubiKey এর শ্রেষ্ঠত্ব এর **Origin Binding** এর মধ্যে নিহিত।

যখন আপনি কোনো সার্ভিসের (যেমন: আপনার ব্যাংক বা ProtonMail) সাথে একটি YubiKey রেজিস্টার করেন, তখন সেই কী (key) নির্দিষ্টভাবে ওই ওয়েব ডোমেইনের জন্য একটি ইউনিক, Asymmetric cryptographic key pair জেনারেট করে। Private key কখনোই YubiKey এর ফিজিক্যাল সিলিকন (physical silicon) থেকে বাইরে বের হয় পণ্ডিত হয় না।

আপনি যখন লগ ইন করার চেষ্টা করেন, তখন ওয়েব ব্রাউজার (Vanadium) YubiKey-এর কাছে একটি চ্যালেঞ্জ (challenge) পাঠায়। লগ ইন রিকোয়েস্ট করা ওয়েবসাইটের এক্সাক্ট ডোমেইনটি YubiKey চেক করে। আপনি যদি বিশ্বাস করেন যে আপনি `paypal.com` এ লগ ইন করছেন, কিন্তু আসলে আপনি `paypa1-secure.com` -এ হোস্ট করা একটি সোফিস্টিকেটেড ফিশিং সাইটে অবস্থান করছেন, তখন YubiKey সেই ডোমেইন মিসম্যাচটি (domain mismatch) ডিটেক্ট করবে।

YubiKey তখন ক্রিপ্টোগ্রাফিক চ্যালেঞ্জটি সাইন (sign) করতে সরাসরি অস্বীকার করে। এটি গাণিতিকভাবে অথেনটিকেশন প্রসেসকে (authentication process) থামিয়ে দেয়। আপনাকে ধোঁকা দিয়ে কোনো ৬-ডিজিটের কোড হাতিয়ে নেয়া সম্ভব নয় কারণ হস্তান্তরের জন্য কোনো কোডই সেখানে থাকে না। একটি YubiKey রিমোট ক্রেডেনশিয়াল ফিশিংকে (remote credential phishing) গাণিতিকভাবে অসম্ভব করে তোলে।

GrapheneOS এ YubiKeys ডেপ্লয় করা

GrapheneOS-কে এমনভাবে ইঞ্জিনিয়ারিং করা হয়েছে যাতে এটি অপারেটিং সিস্টেম লেভেলে হার্ডওয়্যার সিকিউরিটি কীগুলোকে (hardware security keys) নির্বিঘ্নে সাপোর্ট করে, যা আপনার অথেনটিকেশনকে একই সাথে ফ্রিকশনলেস (frictionless) এবং ইমপেনিট্রেবল (impenetrable) বা অভেদ্য করে তোলে।

১. Vanadium Native Support: ওয়েব-ভিত্তিক লগইনগুলোর জন্য, GrapheneOS এর বিল্ট-ইন Vanadium ব্রাউজারে FIDO2 এবং পাসকীগুলোর (passkeys) জন্য নেটিভ Android Credential Manager সাপোর্ট রয়েছে। এটি একটি বিশাল ইঞ্জিনিয়ারিং অ্যাচিভমেন্ট। এর মানে হলো আপনি Sandboxed Google Play কম্প্যাটিবিলিটি লেয়ার (compatibility layer) ইন্সটল করার কোনো প্রয়োজন ছাড়াই সম্পূর্ণ নেটিভভাবে সিকিউর ওয়েব সার্ভিসগুলোতে (যেমন ProtonMail বা আপনার ব্যাংকের ওয়েবসাইট) অথেনটিকেট করার জন্য আপনার YubiKey ব্যবহার করতে পারেন।

২. Sandboxed Google Play (Third-Party Apps): যদিও Vanadium ওয়েব লগইনগুলো নেটিভভাবে পরিচালনা করে, তবে অনেক প্রোপ্রাইটরি থার্ড-পার্টি অ্যাপ্লিকেশন (যেমন নেটিভ ব্যাংকিং অ্যাপ বা কর্পোরেট VPN ক্লায়েন্ট) হার্ডওয়্যার কীগুলোর সাথে কমিউনিকেট করার জন্য স্পষ্টভাবে Google Play Services FIDO APIs-এর ওপর নির্ভর

করে।

আপনি যদি আপনার "Banking/Corporate" কোয়ারেন্টিনড প্রোফাইলের (quarantined profile) (যা সেকশন ২.৬-এ ডিজাইন করা হয়েছে) মধ্যে অ্যাপ্লিকেশন ব্যবহার করে থাকেন, তবে Sandboxed Google Play নির্ভুলভাবে NFC বা USB-C পোর্টের মাধ্যমে অ্যাপ্লিকেশন থেকে ফিজিক্যাল YubiKey-তে FIDO2 ক্রিপ্টোগ্রাফিক রিকোয়েস্টগুলোকে রাউট করে।

Step-by-Step Registration Guide

আপনার অপারেশনাল সিকিউরিটিকে (operational security) সর্বোচ্চ পর্যায়ে উন্নীত করার জন্য, আপনাকে অবশ্যই কমপক্ষে দুটি YubiKey কিনতে হবে (একটি প্রাইমারি কী যা আপনি সাথে বহন করবেন, এবং একটি ব্যাকআপ কী যা কোনো ফিজিক্যাল সেফে সংরক্ষিত থাকবে, যদি প্রাইমারি কীটি হারিয়ে যায় বা নষ্ট হয়ে যায়)।

ফেজ ১: রেজিস্ট্রেশন (Registration) ১. ডেস্কটপ কম্পিউটার বা Vanadium ব্রাউজার ব্যবহার করে একটি হাই-সিকিউরিটি অ্যাকাউন্ট (যেমন: ProtonMail) ওপেন করুন। ২. **Security > Two-Factor Authentication** সেটিংসে নেভিগেট করুন। ৩. **Add Security Key** (বা **FIDO2/WebAuthn**) সিলেক্ট করুন। ৪. যখন নির্দেশ দেওয়া হবে, তখন আপনার GrapheneOS ডিভাইসের USB-C পোর্টে আপনার YubiKey প্রবেশ করান। ৫. **Physical Affirmation:** YubiKey ফ্ল্যাশ করা শুরু করলে এর গোল্ড কন্ট্যাক্ট প্যাডে (gold contact pad) ট্যাপ করুন। এটি প্রমাণ করে যে সেখানে একজন মানুষ শারীরিকভাবে উপস্থিত রয়েছে, যা কোনো রিমোট ম্যালওয়্যারকে (remote malware) সাইলেন্টলি কীটি (key) অ্যাক্টিভেট করা থেকে নিউট্রালাইজ করে। ৬. আপনার ব্যাকআপ YubiKey এর জন্য ঠিক একই প্রসেস রিপিট করুন।

ফেজ ২: NFC এর মাধ্যমে অথেনটিকেশন (Authentication) পরবর্তীতে অ্যাপ্লিকেশনটিতে লগ ইন করার সময়, আপনাকে আপনার সিকিউরিটি কী (security key) চাওয়া হবে। ১. আপনার ইউজারনেম এবং পাসওয়ার্ড এন্টার করুন। ২. যখন সিকিউরিটি কী এর প্রম্পট (prompt) উপস্থিত হবে, তখন আপনার GrapheneOS ডিভাইসের আপার-মিডল ব্যাকে (যেখানে NFC রিডারটি অবস্থিত) YubiKey ট্যাপ করুন। ৩. ক্রিপ্টোগ্রাফিক হ্যান্ডশেক (cryptographic handshake) মিলি-সেকেন্ডের মধ্যে কমপ্লিট হওয়ার জন্য অপেক্ষা করুন, যা ডোমেইন অরিজিনকে (domain origin) ভ্যালিডেট করে এবং আপনাকে সিকিউরভাবে লগ ইন করায়।

The Authentication Layer is Sealed

আপনি এখন আপনার ফোর্ট্রেসের (fortress) গেটগুলোকে ফিশিং-প্রুফ ফিজিক্যাল হার্ডওয়্যার কী (phishing-proof physical hardware keys) দ্বারা আর্মার্ড (armored) করেছেন। আপনার ক্রেডেনশিয়ালগুলো (credentials) আর সোশ্যাল ইঞ্জিনিয়ারিং, SIM swapping বা man-in-the-middle ফিশিং এর প্রতি ভালনারেবল নয়। অথেনটিকেশন লেয়ারটি (authentication layer) এখন গাণিতিকভাবে লকড (locked)।

কিন্তু এই ফোর্ট্রেসটি এখনও সম্পূর্ণ নয়। আমরা অ্যাপ্লিকেশন লেয়ারে ডিভাইসের ভেতরে কী প্রবেশ করে এবং কী বের হয় তা সিকিউর করেছি—এখন ডিভাইসটি ফিজিক্যাল ওয়ার্ল্ডে (physical world) যে অদৃশ্য সিগন্যালগুলো ব্রডকাস্ট করে তা আমাদের সিকিউর করতে হবে। পরবর্তী সেকশনগুলোতে, আমরা আপনার নেটওয়ার্ক ফুটপ্রিন্টকে (network footprint) হার্ডেন (harden) করব: Wi-Fi ট্র্যাকিংকে পরাস্ত করা, আপনার DNS কোয়েরিগুলোকে (DNS queries) এনক্রিপ্ট করা, আপনার VPN টানেলগুলোকে (VPN tunnels) সিল (seal) করা এবং হোস্টাইল সেলুলার ইনফ্রাস্ট্রাকচারকে (hostile cellular infrastructure) নিউট্রালাইজ করা।

দেয়ালগুলো নির্মিত হয়েছে। এখন এয়ারওয়েভস (airwaves) সিকিউর করার সময় এসেছে।

২.১০ Network Anonymity: Wi-Fi Tracking-কে পরাস্ত করা

যদিও আমরা আপনার ডিভাইসের ইন্টারনাল আর্কিটেকচারকে হোস্টাইল সফটওয়্যারের বিরুদ্ধে সিকিউর করেছি, এখন আমাদের এক্সটার্নাল এনভায়রনমেন্টের (external environment) হোস্টাইলিটি মোকাবেলা করতে হবে। আপনি যখন ফিজিক্যাল ওয়ার্ল্ডে চলাফেরা করেন, তখন আপনার স্মার্টফোন একটি ডিজিটাল বীকন (digital beacon) হিসেবে কাজ করে। এটি ক্রমাগত রেডিও সিগন্যাল (radio signals) নির্গত করে, পরিচিত নেটওয়ার্কগুলো খোঁজে, সেলুলার টাওয়ারগুলোর সাথে সংযুক্ত হওয়ার চেষ্টা করে এবং যে কেউ শুনছে তার কাছে নিজের উপস্থিতি ঘোষণা করে।

আধুনিক রিটেইল এবং কর্পোরেট সার্ভেইল্যান্স (surveillance) ইন্ডাস্ট্রি আপনার ফিজিক্যাল মুভমেন্ট ট্র্যাক করার জন্য এই রেডিও সিগন্যালগুলোকে ব্যাপকভাবে ওয়েপোনাইজ (weaponize) করেছে। GrapheneOS কীভাবে এই অদৃশ্য ড্র্যাগনেটকে (dragnet) নিউট্রালাইজ করে তা বোঝার জন্য, আমাদের অবশ্যই Wi-Fi ট্র্যাকিং এর মেকানিক্স (mechanics) ডাইসেক্ট (dissect) করতে হবে।

MAC Tracking এর অ্যানাটমি

আপনার ডিভাইসের প্রতিটি নেটওয়ার্ক ইন্টারফেসে (network interface)—আপনার Wi-Fi রেডিও এবং আপনার Bluetooth রেডিওতে—একটি ইউনিক, ফ্যাক্টরি-অ্যাসাইনড সিরিয়াল নম্বর থাকে যা Media Access Control (MAC) অ্যাড্রেস নামে পরিচিত। গাণিতিকভাবে, এটি একটি পারমানেন্ট আইডেন্টিফায়ার (permanent identifier) যা সরাসরি আপনার ফোনের ফিজিক্যাল সিলিকনের (physical silicon) সাথে আবদ্ধ।

যখন আপনার Wi-Fi অন থাকে, তখন আপনার ডিভাইস "Active Scanning" নামক একটি বিহেভিয়ার (behavior) পারফর্ম করে। এটি ক্রমাগত বাতাসে "Probe Requests" নামক সিগন্যাল ব্রডকাস্ট করে জিজ্ঞেস করে: *"আশেপাশে কি এমন কোনো নেটওয়ার্ক আছে যা আমি চিনি?"*

এই Probe Requests-গুলোর মধ্যে ইনহেরেন্টলি (inherently) আপনার ডিভাইসের MAC অ্যাড্রেস থাকে।

কর্পোরেট সত্তা, শপিং মল, বিমানবন্দর এবং স্মার্ট-সিটি ইনফ্রাস্ট্রাকচারগুলো প্যাসিভ Wi-Fi স্নিফারের (passive Wi-Fi sniffers) নেটওয়ার্ক ডেপ্লয় করে এই বিহেভিয়ারকে এক্সপ্লয়েট (exploit) করে। আপনি যখন কোনো শপিং মলে প্রবেশ করেন, ট্র্যাক হওয়ার জন্য আপনার সক্রিয়ভাবে মলের Wi-Fi নেটওয়ার্কে সংযুক্ত হওয়ার প্রয়োজন নেই। স্নিফারগুলো ব্যাকগ্রাউন্ডে আপনার ফোনের ব্রডকাস্ট করা Probe Requests-গুলো শুনতে থাকে।

আপনি যখন ফুড কোর্টের পাশ দিয়ে হেঁটে যান তখন মল আপনার MAC অ্যাড্রেস লগ করে, যখন আপনি কোনো পোশাকের দোকানে প্রবেশ করেন তখন এটি আবার লগ করে এবং বের হওয়ার সময় তৃতীয়বারের মতো লগ করে। এই ডেটাগুলো অ্যাগ্রিগেট (aggregate) করে, কর্পোরেশন আপনার ফিজিক্যাল লোকেশন, আপনার শপিংয়ের অভ্যাস এবং আপনার পরিদর্শনের সময়কালের একটি নিখুঁত, মিনিট-বাই-মিনিট হিটম্যাপ (heatmap) তৈরি করে। আপনার ফিজিক্যাল বিহেভিয়ারের উপর একটি পারমানেন্ট ডোসিয়ার (permanent dossier) তৈরি করতে তারা আপনার MAC অ্যাড্রেসকে মাসের পর মাস এবং বছরের পর বছর ধরে ট্র্যাক করে।

ফ্লড স্ট্যান্ডার্ড Android ডিফেন্স

MAC ট্র্যাকিংয়ের ভয়াবহ প্রাইভেসি ইমপ্লিকেশন (privacy implications) অনুধাবন করে, কমার্শিয়াল স্মার্টফোন ইন্ডাস্ট্রি একটি ফিক্স (fix) ইমপ্লিমেন্ট করার চেষ্টা করেছিল: **MAC Randomization**।

স্ট্যান্ডার্ড কমার্শিয়াল Android ডিভাইসগুলো এখন "Persistent Per-Network MAC Randomization" নামক একটি প্রোটোকল ব্যবহার করে। সত্যিকারের, ফ্যাক্টরি-অ্যাসাইনড MAC অ্যাড্রেস ব্রডকাস্ট করার পরিবর্তে, অপারেটিং সিস্টেম একটি র‍্যান্ডম, ফেক (fake) MAC অ্যাড্রেস তৈরি করে।

তবে, এই ইমপ্লিমেন্টেশনটি স্ট্রাকচারালি ফ্লড (structurally flawed)। স্ট্যান্ডার্ড Android একটি নির্দিষ্ট নেটওয়ার্কে (যেমন, "Starbucks Wi-Fi") একটি র‍্যান্ডম MAC অ্যাড্রেস অ্যাসাইন করে, কিন্তু এটি সেই র‍্যান্ডম MAC অ্যাড্রেসটি চিরতরে *সংরক্ষণ* (saves) করে রাখে। প্রতিবার যখন আপনি সেই নির্দিষ্ট নেটওয়ার্কে ফিরে যান, তখন ডিভাইসটি ঠিক একই র‍্যান্ডম MAC ব্যবহার করে। তদুপরি, অনেক বিশাল কর্পোরেট চেইন হাজার হাজার লোকেশন জুড়ে একই নেটওয়ার্ক নাম (SSID) ব্যবহার করে।

ফলাফল? স্টারবাকস আপনার *সত্যিকারের* ফ্যাক্টরি MAC অ্যাড্রেস জানে না, তবে তারা এখনও জানে যে এটি *আপনি*। তারা আপনার ফিজিক্যাল অভ্যাসের উপর একটি আনব্রোকেন চেইন অফ সার্ভেইল্যান্স (unbroken chain of surveillance) বজায় রেখে দেশের প্রতিটি স্টারবাকস জুড়ে আপনার "র‍্যান্ডম" MAC অ্যাড্রেস ট্র্যাক করে। এটি একটি প্রাইভেসি থিয়েটার (privacy theater) যা আপনাকে সাধারণ পর্যবেক্ষকদের হাত থেকে রক্ষা করে কিন্তু কর্পোরেট ট্র্যাকিং ইনফ্রাস্ট্রাকচারের বিরুদ্ধে চরমভাবে ব্যর্থ হয়।

GrapheneOS এর মাস্টারস্ট্রোক: Per-Connection Randomization

GrapheneOS ইঞ্জিনিয়াররা বুঝতে পেরেছিলেন যে হাই-সিকিউরিটি এনভায়রনমেন্টের জন্য পারসিস্টেন্ট র‍্যান্ডমাইজেশন (persistent randomization) অপর্যাপ্ত। ফলস্বরূপ, GrapheneOS একটি ব্যাপকভাবে উন্নত, গাণিতিকভাবে আনকম্প্রোমাইজিং (uncompromising) প্রোটোকল ডেপ্লয় করেছে: **Per-Connection MAC Randomization**।

ডিফল্টভাবে এনাবল থাকা এই প্রোটোকলটি আমূল ভিন্ন লজিকের ওপর কাজ করে। প্রতিবার যখন আপনার ডিভাইস কোনো Wi-Fi নেটওয়ার্কে কানেকশন ইনিশিয়েট (initiate) করে, তখন GrapheneOS সম্পূর্ণ নতুন, ক্রিপ্টোগ্রাফিক্যালি র‍্যান্ডম একটি MAC অ্যাড্রেস তৈরি করে।

যদি আপনি সোমবার কোনো ক্যাফের Wi-Fi নেটওয়ার্কে সংযুক্ত হন, তবে আপনার MAC অ্যাড্রেস হলো `A`। যদি আপনি ক্যাফে থেকে চলে যান এবং মঙ্গলবার ফিরে আসেন, তবে আপনার MAC অ্যাড্রেস হবে `B`। যদি আপনি আপনার Wi-Fi ডিসকানেক্ট করেন এবং পাঁচ মিনিট পরে আবার কানেক্ট করেন, তবে আপনার MAC অ্যাড্রেস হবে `C`।

ট্র্যাকিং চেইনটি সম্পূর্ণভাবে ছিন্ন হয়েছে কিনা তা নিশ্চিত করার জন্য, GrapheneOS কোনো নেটওয়ার্কে পুনরায় সংযুক্ত হওয়ার আগে আক্রমণাত্মকভাবে DHCP (Dynamic Host Configuration Protocol) ক্লায়েন্ট স্টেট (client state) ফ্লাশ (flush) করে দেয়। (DHCP হলো সেই প্রোটোকল যা আপনাকে একটি IP address অ্যাসাইন করে; স্ট্যান্ডার্ড ডিভাইসগুলো প্রায়শই তাদের পুরানো IP address-টি ফেরত চায়, যা একটি নতুন MAC অ্যাড্রেস থাকা সত্ত্বেও প্রকাশ করে দেয় যে তারা একই ডিভাইস)। একই সাথে DHCP স্টেট এবং MAC অ্যাড্রেস মুছে ফেলার মাধ্যমে, GrapheneOS নেটওয়ার্কটিকে বাধ্য করে প্রতিবার কানেক্ট হওয়ার সময় আপনাকে সম্পূর্ণ নতুন, আনরেকগনাইজেবল (unrecognizable) ডিভাইস হিসেবে ট্রিট (treat) করতে।

কর্পোরেট সার্ভেইল্যান্স ড্র্যাগনেট সম্পূর্ণভাবে ব্লাইন্ড (blind) হয়ে যায়। তারা আপনার পরিদর্শনের কোনো হিস্টোরিক্যাল প্রোফাইল (historical profile) তৈরি করতে পারেকারি কারণ আপনার ডিভাইসের কোনো হিস্ট্রি নেই।

ব্যাকগ্রাউন্ড স্ক্যানিং এলিমিনেট করা

যেমনটি সেকশন ১.৭-এ আলোচনা করা হয়েছে, প্যাসিভ ট্র্যাকিংকে পরাস্ত করার সবচেয়ে কার্যকর উপায় হলো ব্রডকাস্ট করা পুরোপুরি বন্ধ করে দেওয়া।

স্ট্যান্ডার্ড কমার্শিয়াল ফোনগুলো ব্যাকগ্রাউন্ডে আক্রমণাত্মকভাবে Wi-Fi এবং Bluetooth-এর জন্য স্ক্যান করে, এমনকি কুইক সেটিংস মেন্যুতে টগলগুলো (toggles) "off" দেখালেও। Google-এর নেটওয়ার্ক-ভিত্তিক লোকেশন সার্ভিসগুলোকে (location services) সহায়তা করার জন্য এটি করা হয়।

GrapheneOS ডিফল্টভাবে ব্যাকগ্রাউন্ড স্ক্যানিং সম্পূর্ণ ডিসেবল (disable) করে রাখে। GrapheneOS-এ যখন আপনি আপনার Wi-Fi অফ করেন, তখন রেডিওটি শারীরিকভাবে পাওয়ার ডাউন (power down) হয়ে যায়। কোনো Probe Requests ব্রডকাস্ট হয় না। হাই-রিস্ক ফিজিক্যাল এনভায়রনমেন্টে (যেমন বিমানবন্দর বা কোনো আন্তর্জাতিক সীমান্তে) নেভিগেট করার সময়, একজন এলিট অপারেটর নিশ্চিত করেন যে তার Wi-Fi এবং Bluetooth রেডিওগুলো স্পষ্টভাবে ডিসেবল করা আছে, যা তাদের RF (Radio Frequency) ফুটপ্রিন্টকে সম্পূর্ণ এলিমিনেট করে।

The Enterprise Workaround: MAC Whitelisting

অপারেশনাল সিকিউরিটি প্রায়ই কর্পোরেট এবং এডুকেশনাল নেটওয়ার্কগুলোর রিজিড ইনফ্রাস্ট্রাকচারের (rigid infrastructure) সাথে সাংঘর্ষিক হয়ে দাঁড়ায়।

অনেক এন্টারপ্রাইজ এনভায়রনমেন্ট, বিশ্ববিদ্যালয় এবং হাই-সিকিউরিটি সরকারি ফ্যাসিলিটিগুলো **MAC Whitelisting** নামক একটি নেটওয়ার্ক অ্যাক্সেস কন্ট্রোল মেজার (network access control measure) ব্যবহার করে। তাদের নেটওয়ার্কে সংযুক্ত হওয়ার জন্য, IT ডিপার্টমেন্টের প্রয়োজন হয় তাদের ডেটাবেজে আপনার ডিভাইসের MAC অ্যাড্রেস রেজিস্টার করা। নেটওয়ার্কটি এমন যেকোনো ডিভাইসকে আক্রমণাত্মকভাবে ব্লক করবে যার MAC অ্যাড্রেস অ্যাপ্রুভড লিস্টে (approved list) নেই।

যেহেতু GrapheneOS প্রতিবার কানেক্ট হওয়ার সময় একটি নতুন MAC অ্যাড্রেস জেনারেট করে, তাই একটি GrapheneOS ডিভাইস তাৎক্ষণিকভাবে একটি হোয়াইটলিস্টেড নেটওয়ার্ক (whitelisted network) দ্বারা রিজেক্ট (reject) হবে। যে মুহূর্তে আপনি ডিসকানেক্ট করে পুনরায় কানেক্ট করবেন, আপনার MAC অ্যাড্রেস পরিবর্তন হয়ে যাবে, এবং আপনি আর অ্যাপ্রুভড লিস্টে থাকবেন না।

আপনার অপারেশনাল রিকোয়ারমেন্টস (operational requirements) যদি নির্দেশ করে যে আপনাকে অবশ্যই একটি হোয়াইটলিস্টেড নেটওয়ার্কে সংযুক্ত হতে হবে, তবে GrapheneOS একটি ট্যাকটিক্যাল ওয়ার্কঅ্যারাউন্ড (tactical workaround) প্রদান করে।

১. **Settings > Network & internet > Internet**-এ নেভিগেট করুন। ২. নির্দিষ্ট সেভ করা নেটওয়ার্কটি (যেমন: আপনার কর্পোরেট Wi-Fi) সিলেক্ট করুন। ৩. **Privacy** অপশনে ট্যাপ করুন। ৪. সেটিংটি **Use randomized MAC (default)** থেকে **Use per-network randomized MAC**-এ পরিবর্তন করুন।

এই অপশনটি সিলেক্ট করার মাধ্যমে, আপনি কেবল ওই *নির্দিষ্ট নেটওয়ার্কটির জন্যই* সিকিউরিটি ডাউনগ্রেড (downgrade) করছেন। GrapheneOS একটি র‍্যান্ডম MAC অ্যাড্রেস জেনারেট করবে, কিন্তু এটি সেভ করে রাখবে এবং প্রতিবার সেই নির্দিষ্ট কর্পোরেট নেটওয়ার্কে কানেক্ট করার সময় ধারাবাহিকভাবে এটি ব্যবহার করবে, যা আপনাকে IT ডিপার্টমেন্টের কাছে MAC অ্যাড্রেসটি রেজিস্টার করার সুযোগ দেয়।

> ***OpSec Warning:*** *কখনোই "Use device MAC" অপশনটি সিলেক্ট করবেন করবেন না, যদি না কোনো হোস্টাইল IT ডিপার্টমেন্ট দ্বারা একেবারে বাধ্যতামূলক করা হয় যারা সক্রিয়ভাবে র‍্যান্ডমাইজড MAC রেঞ্জগুলো (randomized MAC ranges) ব্লক করে। আপনার সত্যিকারের, ফ্যাক্টরি-অ্যাসাইনড হার্ডওয়্যার MAC অ্যাড্রেস প্রদান করা ডিভাইসের সিলিকনের ফিজিক্যাল অ্যানোনিমিটিকে (physical anonymity) স্থায়ীভাবে আপস (compromise) করে।*

দ্য আল্টিমেট থ্রেট: ওয়াইফাই সেন্সিং (CSI) এবং ফিজিক্যাল ট্র্যাকিং

MAC র‍্যান্ডমাইজেশন আপনাকে ডিজিটাল ট্র্যাকিং থেকে বাঁচায়, কিন্তু ফিজিক্যাল বা সিগন্যাল ট্র্যাকিং থেকে নয়। বর্তমান সিগন্যাল ইন্টেলিজেন্স (SIGINT) জগতে **ওয়াইফাই সেন্সিং (Wi-Fi Sensing বা CSI)** নামক একটি ভয়াবহ প্রযুক্তি রয়েছে (যেমন: *WiFiDensePose (https://github.com/rokibulroni/WiFiDensePose)* প্রজেক্ট)। এই প্রযুক্তিগুলো কোনো ফোন হ্যাক করে না, বরং সাধারণ 2.4 GHz ওয়াইফাই রাউটারের সিগন্যাল ব্যবহার করে দেওয়ালের ওপার থেকে রিয়েল-টাইমে মানুষের ফিজিক্যাল পোজ (বসা, দাঁড়ানো), মুভমেন্ট এবং শ্বাস-প্রশ্বাস ডিটেক্ট করতে পারে। যখন ওয়াইফাইয়ের রেডিও তরঙ্গ আপনার শরীরের সাথে ধাক্কা খায়, তখন সিগন্যালে তৈরি হওয়া অতি ক্ষুদ্র বিকৃতি (Channel State Information) বিশ্লেষণ করে আর্টিফিশিয়াল ইন্টেলিজেন্স আপনার শরীরের একটি থ্রিডি ম্যাপ তৈরি করে ফেলে।

কীভাবে নিজেকে ডিফেন্ড করবেন: ১. ISP রাউটার বর্জন: আপনার ইন্টারনেট প্রোভাইডারের দেওয়া রাউটারে তারা ফার্মওয়্যার আপডেট করে সহজেই CSI-ট্র্যাকিং মডিউল পুশ করতে পারে। সবসময় নিজস্ব কেনা রাউটার ব্যবহার করুন এবং সম্ভব হলে OpenWRT-এর মতো ওপেন সোর্স ফার্মওয়্যার ইনস্টল করুন। ২. **RF আইসোলেশন:** আপনার থ্রেট মডেল যদি এক্সট্রিম হয় (যেমন: রাষ্ট্রীয় পর্যায়ের নজরদারি), তবে ফ্যারাডে কেজ (Faraday Cage) বা ব্রডব্যান্ড রেডিও ফ্রিকোয়েন্সি জ্যামার ব্যবহার করা ছাড়া এই ফিজিক্যাল রাডার থেকে পালানোর কোনো সফটওয়্যার-ভিত্তিক উপায় নেই।

আপনার রেডিও ট্রান্সমিশনের ওপর কঠোর নিয়ন্ত্রণ এবং per-connection randomization-এর গাণিতিক অ্যানোনিমিটির মাধ্যমে আপনি নেটওয়ার্ক লেভেলের ট্র্যাকিং পরাস্ত করতে পারবেন। কিন্তু মনে রাখবেন—যেটিকে আইডেন্টিফাই করা যায় না, নেটওয়ার্ক তাকে ট্র্যাক করতে পারে না, তবে রেডিও ওয়েভ আপনার ফিজিক্যাল উপস্থিতিকে ঠিকই অনুভব করতে পারে।

২.১১ The Encrypted Tunnel: VPNs এবং Kill Switches

আপনি যখন কোনো পাবলিক Wi-Fi নেটওয়ার্কে সংযুক্ত হন—তা বিমানবন্দরে, হোটেলে বা কফি শপেই হোক না কেন—আপনি স্বেচ্ছায় আপনার হার্ডওয়্যারটিকে সরাসরি একটি আনট্রাস্টেড (untrusted), সম্ভাব্য হোস্টাইল ইনফ্রাস্ট্রাকচারে (hostile infrastructure) প্লাগ করছেন।

রাউটার কে নিয়ন্ত্রণ করছে তা দেখার কোনো সুযোগ আপনার নেই। আপনার ডিভাইস থেকে বের হওয়া প্রতিটি আনএনক্রিপ্টেড বাইট ডেটা (unencrypted byte of data) মনিটর করার ক্ষমতা নেটওয়ার্ক অ্যাডমিনিস্ট্রেটরের রয়েছে। এমনকি আপনি যদি সিকিউর `HTTPS` ওয়েবসাইটগুলো ভিজিট করেন, তবুও নেটওয়ার্ক অ্যাডমিনিস্ট্রেটর ট্রাফিক অ্যানালাইসিস (traffic analysis) এক্সিকিউট করতে পারে, আপনি কোন নির্দিষ্ট ডোমেইন নামগুলো ভিজিট করেছেন এবং ঠিক কোন সময়ে ভিজিট করেছেন তা লগ (log) করতে পারে। আরো বেশি অ্যাগ্রেসিভ সিনারিওতে, আপনার এনক্রিপশন স্ট্রিপ (strip) করে আপনার ক্রেডেনশিয়ালগুলো (credentials) হারভেস্ট (harvest) করার প্রচেষ্টায়, অ্যাডভারসারিরা (adversaries) Man-in-the-Middle (MitM) অ্যাটাক এক্সিকিউট করার জন্য ডিজাইন করা ম্যালিশিয়াস (malicious) "evil twin" অ্যাক্সেস পয়েন্ট সেটআপ করতে পারে।

একটি হোস্টাইল নেটওয়ার্ক এনভায়রনমেন্টে সিকিউরভাবে অপারেট করতে, অ্যাডভারসারির ইনফ্রাস্ট্রাকচারের মধ্য দিয়ে আপনাকে অবশ্যই একটি ক্রিপ্টোগ্রাফিক্যালি সিকিউর টানেল তৈরি করতে হবে। আপনাকে অবশ্যই একটি **Virtual Private Network (VPN)** ডেপ্লয় (deploy) করতে হবে।

The WireGuard Standard

কয়েক দশক ধরে, সাইবারসিকিউরিটি ইন্ডাস্ট্রি OpenVPN এবং IPsec প্রোটোকলের ওপর নির্ভরশীল ছিল। গাণিতিকভাবে সিকিউর হলেও, এই লেগ্যাসি প্রোটোকলগুলো (legacy protocols) অবিশ্বাস্যভাবে ভারী। এগুলো মোবাইলের ব্যাটারি ড্রেইন (drain) করে এবং সেলুলার টাওয়ার থেকে Wi-Fi নেটওয়ার্কে ট্রান্সিশন করার সময় মারাত্মক ল্যাটেন্সির (latency) শিকার হয়।

এলিট মোবাইল অপারেটররা প্রায় একচেটিয়াভাবে **WireGuard** প্রোটোকলে ট্রান্সিশন করেছে। WireGuard হলো একটি আধুনিক, ব্যাপকভাবে সিম্পলিফাইড VPN প্রোটোকল যা স্টেট-অফ-দ্য-আর্ট ক্রিপ্টোগ্রাফির (যেমন: Noise protocol framework এবং Curve25519) ওপর তৈরি। যেহেতু এটি মাত্র কয়েক হাজার লাইনের কোড নিয়ে গঠিত, তাই এটি এক্সসেপশনালি (exceptionally) ফাস্ট, গাণিতিকভাবে অডিটেবল (auditable) এবং আপনার ডিভাইস যখন নেটওয়ার্ক পরিবর্তন করে তখন এটি মুহূর্তের মধ্যে রিকানেক্ট (reconnect) হয়।

কোনো VPN প্রোভাইডার সিলেক্ট করার সময়, তাদেরকেই অগ্রাধিকার দিন যারা নেটিভ WireGuard সাপোর্ট দেয়, প্রাইভেসি-রেসপেক্টিং জুরিসডিকশনে (privacy-respecting jurisdictions) অপারেট করে এবং কঠোর নো-লগিং পলিসি (no-logging policies) এনফোর্স (enforce) করে। **Mullvad VPN** এবং **ProtonVPN** বর্তমানে হাই-সিকিউরিটি কমিউনিটির মধ্যে গোল্ড স্ট্যান্ডার্ড (gold standards) হিসেবে স্বীকৃত।

The Mechanism of the Kill Switch

একটি হাই-কোয়ালিটি VPN ইন্সটল করা হলো কেবল প্রথম পদক্ষেপ। একটি VPN এর আসল ভালনারেবিলিটি (vulnerability) নির্ভর করে তখন কী ঘটে যখন এনক্রিপ্টেড টানেলটি ফেইল (fail) করে।

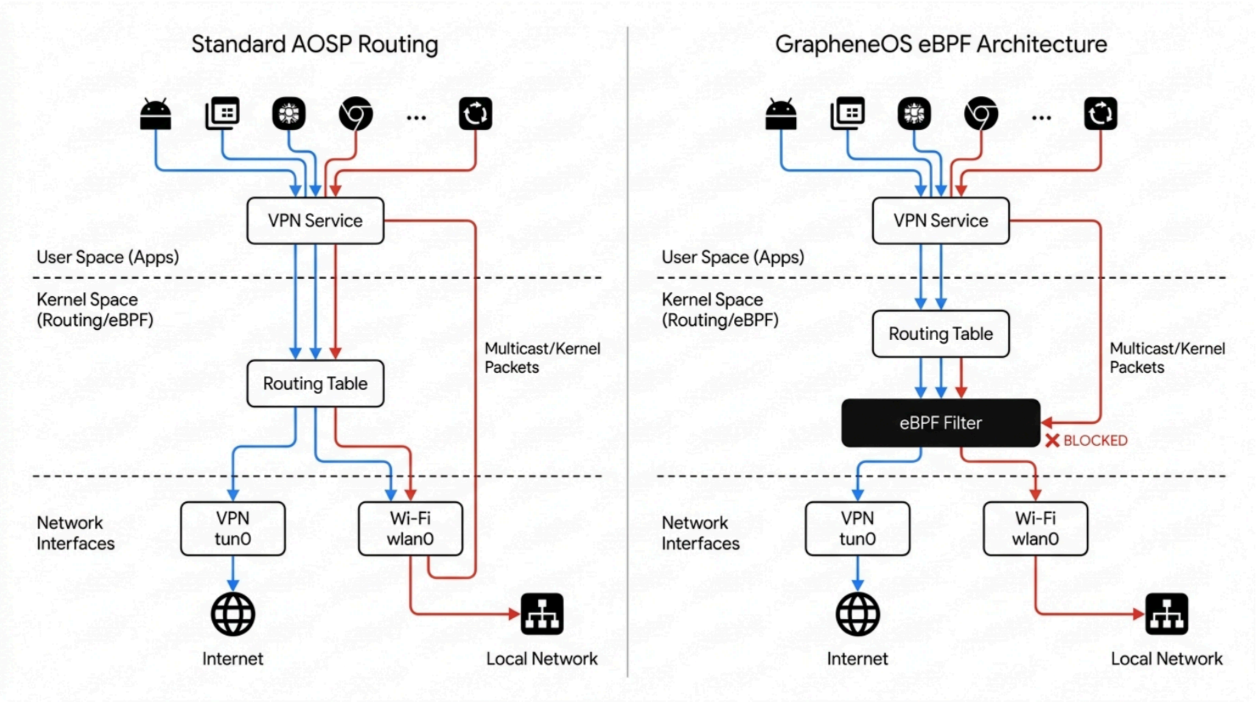
যদি আপনার VPN সার্ভার রিবুট (reboot) হয়, বা যদি আপনার সেলুলার সিগন্যাল মুহূর্তের জন্য ড্রপ করে, তবে VPN কানেকশনটি বিচ্ছিন্ন হয়ে যাবে। একটি পোরলি-কনফিগার করা (poorly configured) ডিভাইসে, অপারেটিং সিস্টেম সাইলেন্টলি স্ট্যান্ডার্ড, আনএনক্রিপ্টেড নেটওয়ার্ক রাউটিংয়ে ডিফল্ট ব্যাক (default back) করবে। টানেলটি যে কলাপ্স (collapse) করেছে তা বুঝতে না পেরে, আপনার অ্যাপ্লিকেশনগুলো হোস্টাইল নেটওয়ার্কের মধ্য দিয়ে প্লেইন টেক্সট (plain text) হিসেবে সেনসিটিভ ডেটা ট্রান্সমিট করতে থাকবে।

এটি মিটিগেট (mitigate) করার জন্য, আপনাকে অবশ্যই **Kill Switch** এনগেজ (engage) করতে হবে।

এটি এনফোর্স (enforce) করার জন্য GrapheneOS নেটিভ Android VPN রাউটিং আর্কিটেকচার লিভারেজ (leverage) করে। একবার আপনি আপনার VPN অ্যাপ্লিকেশন ইন্সটল এবং কানেক্ট করার পর, অপারেটিং সিস্টেমটিকে একটি রুথলেস গেটকিপার (ruthless gatekeeper) হিসেবে কাজ করার জন্য আপনাকে অবশ্যই কনফিগার করতে হবে।

১. **Settings > Network & internet > VPN**-এ নেভিগেট করুন। ২. আপনার নির্দিষ্ট VPN অ্যাপ্লিকেশনের পাশের সেটিংস গিয়ারটিতে (settings gear) ট্যাপ করুন। ৩. **Always-on VPN** এনাবল করুন। (এটি বুট (boot) বা নেটওয়ার্ক পরিবর্তনের সময় অটোমেটিক্যালি টানেলটি রি-এস্টাব্লিশ (re-establish) করার জন্য OS-কে নির্দেশ দেয়)। ৪. **Block connections without VPN** এনাবল করুন।

eBPF Enforcement of Cryptographic Tunnel Integrity



The integration of an Extended Berkeley Packet Filter (eBPF) within the kernel space provides deep packet inspection, actively dropping rogue multicast and kernel-generated packets that bypass standard Android routing tables.

এই ফাইনাল টগলটিই (toggle) হলো কিল সুইচ (Kill Switch)। এটি এনগেজ (engage) করার পর, অপারেটিং সিস্টেম একটি কঠোর ফায়ারওয়াল রুল (firewall rule) এক্সিকিউট করে: যদি VPN টানেলটি অ্যাক্টিভ এবং ভেরিফায়েড (verified) না থাকে, তবে কার্নেল লেভেলে (kernel level) সমস্ত ইনবাউন্ড এবং আউটবাউন্ড নেটওয়ার্ক ট্রাফিক ব্লক হয়ে যায়। ক্রিপ্টোগ্রাফিক টানেলটি রিস্টোর (restore) না হওয়া পর্যন্ত এক বাইট ডেটাও ডিভাইসে প্রবেশ করতে বা বের হতে পারে না।

> 💡 ***Pro Tip:*** *সর্বদা "Always-on VPN" এবং "Block connections without VPN"—এই **দুটিই** একসাথে এনাবল করবেন। শুধুমাত্র প্রথম টগলটি অটো-রিকানেক্ট (auto-reconnect) করে, এটি রিকানেকশন উইন্ডোর (reconnection window) সময় ডেটা লিক হওয়া আটকাতে পারে ঘন ঘন। কেবলমাত্র দ্বিতীয় টগলটি একটি হার্ড কার্নেল-লেভেল ব্লক এনফোর্স করে, যা আনএনক্রিপ্টেড ডেটাকে বেরিয়ে যেতে বাধা দেয়।*

দ্য স্ট্যান্ডার্ড Android লিক ফ্ল (The Standard Android Leak Flaw)

আপনি যদি কোনো স্ট্যান্ডার্ড কমার্শিয়াল Android ডিভাইসে "Block connections without VPN" টগলটি এনাবল করেন, তবে আপনার মনে হতে পারে যে আপনার ১০০% ট্রাফিক এখন সিকিউরড (secured)। কিন্তু আপনি মারাত্মকভাবে ভুল করবেন।

VPN রাউটিংয়ের ক্ষেত্রে স্ট্যান্ডার্ড Android-এ একটি বড় আর্কিটেকচারাল ফ্ল (architectural flaw) রয়েছে। অপারেটিং সিস্টেম ইচ্ছাকৃতভাবেই নির্দিষ্ট "সিস্টেম" ফাংশনগুলোর জন্য কিল সুইচকে বাইপাস করার সুযোগ করে দেয়।

একটি স্ট্যান্ডার্ড Google Pixel বা Samsung Galaxy ডিভাইসে কিল সুইচ পুরোপুরি এনগেজ থাকা সত্ত্বেও, অপারেটিং সিস্টেম মাল্টিকাস্ট নেটওয়ার্ক প্যাকেটগুলোকে (multicast network packets) VPN টানেল বাইপাস করার অনুমতি দেয়। এর চেয়েও খারাপ বিষয় হলো, যদি VPN অ্যাপটি কিছুক্ষণের জন্য ক্র্যাশ করে, তবে স্ট্যান্ডার্ড Android লোকাল নেটওয়ার্কে DNS কোয়েরি (DNS queries—যে রিকোয়েস্টগুলো ডোমেইন নেমকে IP অ্যাড্রেসে রূপান্তর করে) লিক হতে দেয়। সবশেষে, Android তার নেটিভ "কানেক্টিভিটি চেকস" বা connectivity checks (ইন্টারনেট কাজ করছে কি না তা দেখার জন্য গুগল সার্ভারে পিং করা)-কে পুরোপুরি VPN বাইপাস করার অনুমতি দেয়।

আপনি যদি কোনো হোস্টাইল এয়ারপোর্ট Wi-Fi-এ কানেক্ট করেন, তবে নেটওয়ার্ক অ্যাডমিনিস্ট্রেটর দেখতে পাবে যে আপনার ফোনটি VPN টানেলের বাইরে গুগলের সার্ভারে পিং (ping) করছে। এর ফলে আপনার আসল IP অ্যাড্রেস এবং আপনি যে একটি Android ডিভাইস চালাচ্ছেন—তা তাৎক্ষণিকভাবে এক্সপোজড (exposed) হয়ে যায়।

দ্য GrapheneOS হার্ডেনিং (The GrapheneOS Hardening)

GrapheneOS-এর মতে, VPN বাইপাস করে বেরিয়ে যাওয়া যেকোনো ট্রাফিকই হলো একটি ক্যাটাস্ট্রফিক সিকিউরিটি ফেইলিওর (catastrophic security failure)।

এটি নিউট্রালাইজ (neutralize) করার জন্য, GrapheneOS ইঞ্জিনিয়াররা Android নেটওয়ার্কিং স্ট্যাকে (networking stack) বিশাল পরিবর্তন এনেছেন। প্রথমত, GrapheneOS স্ট্যান্ডার্ড eBPF (Extended Berkeley Packet Filter) ফায়ারওয়ালকে এমনভাবে এক্সটেন্ড (extend) করেছে, যাতে এটি সব ধরনের মাল্টিকাস্ট প্যাকেট বাইপাস সম্পূর্ণভাবে ব্লক করতে পারে। দ্বিতীয়ত, VPN অ্যাপ্লিকেশন ক্র্যাশ করলে যে রেস কন্ডিশনের (race condition) সৃষ্টি হয়, সেই সময় ইউনিকাস্ট (unicast) DNS লিক পুরোপুরি প্রতিরোধ করে।

এর পাশাপাশি, GrapheneOS আপনাকে সর্বশেষ লিক—কানেক্টিভিটি চেকস (connectivity checks) সিল (seal) করারও সুযোগ দেয়।

ডিফল্টভাবে, GrapheneOS গুগলের কানেক্টিভিটি চেক সার্ভারগুলোকে তার নিজস্ব প্রাইভেসি-রেস্পেক্টিং সার্ভার দিয়ে রিপ্লেস করে দেয়। যাইহোক, সম্পূর্ণ ১০০% এনক্রিপ্টেড টানেলিং (encrypted tunneling) অর্জন করার জন্য, আপনি ইন্টারনেট কানেক্টিভিটি চেক করা পুরোপুরি বন্ধ করে দিতে অপারেটিং সিস্টেমকে নির্দেশ দিতে পারেন।

১. **Settings** > **Network & internet** > **Internet**-এ নেভিগেট করুন। ২. **Network preferences**-এ ট্যাপ করুন। ৩. **Internet connectivity checks** টগলটি বন্ধ করুন।

যখন আপনি GrapheneOS নেটওয়ার্ক হার্ডেনিং এবং VPN কিল সুইচের পাশাপাশি এই সর্বশেষ স্টেপটি এক্সিকিউট করবেন, তখন আপনি পারফেক্ট নেটওয়ার্ক আইসোলেশনের (perfect network isolation) একটি স্টেট (state) অর্জন করবেন।

আপনার ডিভাইস থেকে শূন্য ডেটা (zero data)—কোনো DNS কোয়েরি নয়, কোনো ক্যাপটিভ পোর্টাল পিং (captive portal ping) নয়, কোনো মাল্টিকাস্ট প্যাকেট নয় এবং কোনো অ্যাপ টেলিমেট্রি নয়—বেরিয়ে যেতে পারবে না, যতক্ষণ না এটি হেভিলি এনক্রিপ্টেড (heavily encrypted) হয়ে সুরক্ষিতভাবে WireGuard টানেলের মধ্য দিয়ে রাউট (route) করা হচ্ছে। আপনি সফলভাবে অ্যাডভারসারির বিরুদ্ধে নেটওয়ার্ককে ওয়েপনাইজ (weaponize) করেছেন।

২.১২ Private DNS: Encrypting the Phone Book

আধুনিক ইন্টারনেট ব্যবহারকারীদের মধ্যে একটি সাধারণ ভুল ধারণা রয়েছে যে `HTTPS` সম্পূর্ণ অ্যানোনিমিটি (anonymity) প্রদান করে। যখন আপনি আপনার ব্রাউজারের অ্যাড্রেস বারে প্যাডলক আইকনটি (padlock icon) দেখতে পান যা একটি সিকিউর `HTTPS` কানেকশন নির্দেশ করে, তখন ট্রাফিকের *কন্টেন্ট* (content) সত্যিই এনক্রিপ্টেড (encrypted) থাকে। আপনি যদি কোনো ওয়েবসাইটে কোনো মেসেজ বা পাসওয়ার্ড পাঠান, তবে কোনো ইভসড্রপার (eavesdropper) তা পড়তে পারবে না।

তবে, মেটাডেটার (metadata) ক্ষেত্রে `HTTPS`-এ একটি মারাত্মক ভালনারেবিলিটি (vulnerability) রয়েছে। যদিও অ্যাটাকার দেখতে পায় না যে আপনি কোনো ওয়েবসাইটে *কী* করছেন, ইন্টারনেটের লেগ্যাসি আর্কিটেকচার (legacy architecture) তাদেরকে ঠিক *কোন* ওয়েবসাইটগুলো আপনি ভিজিট করছেন তা দেখার সুযোগ করে দেয়।

এই ইন্টেলিজেন্স লিকটি (intelligence leak) Domain Name System (DNS) এর কারণে ঘটে।

The Unencrypted Phone Book

কম্পিউটার `proton.me` বা `grapheneos.org`-এর মতো ডোমেইন নামগুলো (domain names) বোঝে না। তারা কেবল IP addresses বোঝে, যেগুলো `185.70.42.37` এর মতো সংখ্যার একটি স্ট্রিং (strings of numbers)।

এই গ্যাপটি (gap) দূর করতে ইন্টারনেট DNS এর ওপর নির্ভর করে। DNS একটি গ্লোবাল, ডিজিটাল ফোন বুক (digital phone book) হিসেবে কাজ করে। আপনি যখন Vanadium-এ `proton.me` টাইপ করেন, তখন আপনার ডিভাইস একটি DNS query পাঠায় যেখানে জিজ্ঞেস করা হয়, "`proton.me`-এর IP address কী?" একটি DNS সার্ভার রিকোয়েস্টটি (request) গ্রহণ করে, নম্বরটি খুঁজে বের করে এবং এটি আপনার ডিভাইসে ফেরত দেয় যাতে কানেকশনটি স্থাপন করা যায়।

ঐতিহাসিকভাবে, এই সম্পূর্ণ প্রসেসটি **প্লেইন টেক্সট**-এ (plain text) ঘটে থাকে।

আপনি যদি কোনো VPN ব্যবহার না করেন, তবে আপনার DNS রিকোয়েস্টগুলো (DNS requests) স্বয়ংক্রিয়ভাবে আপনার Internet Service Provider (ISP) বা লোকাল Wi-Fi নেটওয়ার্ক অ্যাডমিনিস্ট্রেটর দ্বারা প্রদত্ত ডিফল্ট সার্ভারগুলোর মাধ্যমে রাউট (route) করা হয়। যেহেতু এই রিকোয়েস্টগুলো আনএনক্রিপ্টেড (unencrypted) থাকে, আপনার ISP আপনার ভিজিট করা প্রতিটি ওয়েবসাইটের একটি সম্পূর্ণ, সেকেন্ড-বাই-সেকেন্ড হিস্ট্রি (history) মনিটর (monitor), লগ (log) এবং মনিটাইজ (monetize) করতে পারে। তদুপরি, একটি হোস্টাইল পাবলিক Wi-Fi নেটওয়ার্কে,

একজন ম্যালিশিয়াস অ্যাক্টর (malicious actor) আপনার প্লেইন টেক্সট DNS কোয়েরিকে ইন্টারসেপ্ট (intercept) করতে পারে এবং একটি "DNS Spoofing" অ্যাটাক এক্সিকিউট করতে পারে—ইচ্ছাকৃতভাবে আসল ওয়েবসাইটের পরিবর্তে আপনাকে একটি ফেক ফিশিং সার্ভারের (fake phishing server) IP address প্রদান করে।

অ্যাবসলিউট (absolute) নেটওয়ার্ক সিকিউরিটি অর্জন করতে হলে, আমাদের অবশ্যই এই ফোন বুকটিকে এনক্রিপ্ট করতে হবে।

ক্রিপ্টোগ্রাফিক ফিক্স: Private DNS (DoT)

DNS সার্ভেইল্যান্স (surveillance) নিউট্রালাইজ (neutralize) করার জন্য, সাইবারসিকিউরিটি ইন্ডাস্ট্রি **DNS-over-TLS (DoT)** ডেভেলপ করেছে।

প্লেইন টেক্সটে পুরো নেটওয়ার্ক জুড়ে আপনার DNS কোয়েরিগুলো (queries) চিৎকার করে জানানোর পরিবর্তে, DNS-over-TLS কোয়েরিটিকে একটি হেভিলি (heavily) এনক্রিপ্টেড ক্রিপ্টোগ্রাফিক টানেলের (cryptographic tunnel) ভেতরে র‍্যাপ (wrap) করে। আপনি যখন কোনো ওয়েবসাইটের IP address এর জন্য রিকোয়েস্ট করেন, তখন লোকাল নেটওয়ার্ক অ্যাডমিনিস্ট্রেটর এবং আপনার ISP এনক্রিপ্টেড জিব্বারিশের (encrypted gibberish) একটি স্ট্রিম ছাড়া আর কিছুই দেখতে পায় না। তারা আর আপনার ওয়েব হিস্ট্রি লগ করতে পারবে না এবং আপনার ট্রাফিককে রিডাইরেক্ট (redirect) করার জন্য তারা রেজাল্টগুলোকেও স্পুফ (spoof) করতে পারবে না।

GrapheneOS অপারেটিং সিস্টেমের ডিপেস্ট লেভেলে (deepest level) **Private DNS** নামক একটি ফিচারের অধীনে নেটিভভাবে এই প্রোটোকলটিকে সাপোর্ট করে।

আর্কিটেক্টিং লেয়ারড ডিফেন্স

আপনি হয়তো জিজ্ঞেস করতে পারেন: *আমি যদি ইতিমধ্যেই একটি WireGuard VPN ব্যবহার করে থাকি (যেমন সেকশন ২.১১-এ বিস্তারিত বলা হয়েছে), তবে কি আমাকে Private DNS কনফিগার করতে হবে?*

এলিট অপারেশনাল সিকিউরিটিতে (operational security), এর উত্তর সর্বদাই হ্যাঁ। এটি হলো **Defense-in-Depth** এর ডকট্রিন (doctrine)।

আপনি যখন একটি হাই-কোয়ালিটি VPN ব্যবহার করেন, তখন আপনার DNS কোয়েরিগুলো সাধারণত VPN প্রোভাইডারের ইন্টারনাল DNS সার্ভারগুলো দ্বারা সিকিউরভাবে হ্যান্ডেল (handle) করা হয়। তবে, VPN ক্র্যাশ (crash) করতে পারে। আপনি হয়তো এমন কোনো হাইপার-অ্যাগ্রেসিভ (hyper-aggressive) কর্পোরেট ফায়ারওয়ালের (corporate firewall) সম্মুখীন হতে পারেন যা সক্রিয়ভাবে WireGuard ট্রাফিক ব্লক করে, যা আপনাকে কোনো ক্রিটিক্যাল লোকাল সার্ভিসে (local service) অ্যাক্সেস করার জন্য সাময়িকভাবে VPN ডিসেবল (disable) করতে বাধ্য করে।

যদি আপনি একটি Private DNS ফলব্যাক (fallback) ছাড়াই আপনার VPN ডিসেবল করেন, তবে আপনার ডিভাইসটি তাৎক্ষণিকভাবে লোকাল, হোস্টাইল, আনএনক্রিপ্টেড DNS সার্ভারগুলো ব্যবহার করায় রিভার্ট (revert) করবে। অপারেটিং সিস্টেম লেভেলে Private DNS কনফিগার করার মাধ্যমে, আপনি একটি আনব্রেকেবল সেকেন্ডারি বাল্কহেড (unbreakable secondary bulkhead) এস্টাব্লিশ (establish) করেন। এমনকি যদি VPN টানেলটি কলাপ্স (collapse) করে, তবুও আপনার DNS কোয়েরিগুলো ISP-র হাত থেকে ক্রিপ্টোগ্রাফিক্যালি শিল্ডেড (shielded) থাকে।

GrapheneOS-এ Private DNS কনফিগার করা

এই ডিফেন্সটি ডেপ্লয় (deploy) করার জন্য, আপনাকে অবশ্যই স্পষ্টভাবে GrapheneOS-কে নির্দেশ দিতে হবে যাতে এটি লোকাল নেটওয়ার্কের ডিফল্ট DNS-কে বাইপাস (bypass) করে এবং আপনার এনক্রিপ্টেড কোয়েরিগুলোকে একটি ট্রাস্টেড (trusted), হাই-সিকিউরিটি DNS প্রোভাইডারের কাছে রাউট (route) করে।

রিকমেন্ডেড প্রোভাইডারস (Recommended Providers):

- **Quad9:** সুইজারল্যান্ড-ভিত্তিক একটি ইনডিপেনডেন্ট নন-প্রফিট (independent non-profit)। Quad9 কঠোর প্রাইভেসি পলিসি এনফোর্স করে, আপনার IP address লগ করে না এবং পরিচিত ম্যালিশিয়াস ডোমেইন (malicious domains) ও ম্যালওয়্যার নেটওয়ার্কগুলোতে (malware networks) পাঠানো DNS কোয়েরিগুলোকে অটোমেটিক্যালি ব্লক করে।
- **Cloudflare (1.1.1.1):** একটি বিশাল কর্পোরেট সত্তা (corporate entity) হওয়া সত্ত্বেও, Cloudflare ২৪ ঘণ্টার মধ্যে লগগুলো পার্জ (purge) করার পাবলিক কমিটমেন্ট (public commitments) সহ একটি অবিশ্বাস্য ফাস্ট এবং হাইলি রিলায়েবল (highly reliable) এনক্রিপ্টেড DNS সার্ভিস প্রদান করে।

কনফিগারেশন প্রোটোকল (The Configuration Protocol): ১. **Settings > Network & internet > Private DNS**-এ নেভিগেট করুন। ২. সেটিংটি **Automatic** থেকে ম্যানুয়াল এনফোর্সমেন্টে (manual enforcement) পরিবর্তন করুন। ৩. **Private DNS provider hostname** সিলেক্ট করুন। ৪. আপনার নির্বাচিত সিকিউর প্রোভাইডারের হোস্টনেমটি (hostname) এন্টার করুন:

- Quad9 এর জন্য, এন্টার করুন: `dns.quad9.net`
- Cloudflare এর জন্য, এন্টার করুন: `1dot1dot1dot1.cloudflare-dns.com` ৫. **Save**-এ ট্যাপ করুন।

যে মুহূর্তে আপনি Save-এ ট্যাপ করবেন, GrapheneOS প্রোভাইডারের সাথে একটি সিকিউর TLS হ্যান্ডশেক (secure TLS handshake) ইনিশিয়েট (initiate) করবে। এখন থেকে, আপনার সম্পূর্ণ ডিভাইস জুড়ে প্রতিটি ডোমেইন রেজোলিউশন (domain resolution)—তা Vanadium ব্রাউজার, কোনো ব্যাকগ্রাউন্ড অ্যাপ বা স্বয়ং অপারেটিং সিস্টেম দ্বারাই রিকোয়েস্ট করা হোক না কেন—এনক্রিপ্টেড হবে।

আপনি সফলভাবে ISP-গুলোকে ব্লাইন্ড (blind) করেছেন। আপনি লোকাল DNS স্পুফিং (local DNS spoofing) নিউট্রালাইজ করেছেন। MAC র‍্যান্ডমাইজেশন (MAC randomization) এবং WireGuard কিল সুইচ (kill switch)-এর সাথে সংযুক্ত হয়ে, আপনার এক্সটার্নাল নেটওয়ার্ক ফুটপ্রিন্ট (external network footprint) এখন সম্পূর্ণভাবে ক্রিপ্টোগ্রাফিক্যালি সিল্ড (sealed)।

২.১৩ Baseband Warfare: LTE-Only Mode এবং Stingrays

এতক্ষণ পর্যন্ত আমরা এই অনুমানের ওপর ভিত্তি করে কাজ করেছি যে, আমরা যেসব নেটওয়ার্কে যুক্ত হচ্ছি—যেমন Wi-Fi রাউটার, হোম নেটওয়ার্ক এবং কমার্শিয়াল সেলুলার টাওয়ারগুলো—সেগুলো আনট্রাস্টেড হলেও, অন্তত বৈধ (legitimate)। কিন্তু এখন আমাদের থ্রেট মডেল (threat model)-কে আরও এক ধাপ ওপরে নিয়ে যেতে হবে। এমন একটি ভয়াবহ পরিস্থিতির মুখোমুখি হতে হবে যেখানে স্বয়ং ফিজিক্যাল ইনফ্রাস্ট্রাকচারটিই প্রতিকূল, প্রতারণামূলক এবং সুনির্দিষ্টভাবে আপনার বিরুদ্ধে রেডিও ওয়েভ বা এয়ারওয়েভকে অস্ত্র হিসেবে ব্যবহার করার জন্য ডিজাইন করা।

এই থ্রেট বা হুমকিটি ভালোভাবে বুঝতে হলে আমাদের আধুনিক স্মার্টফোনের অভ্যন্তরীণ হার্ডওয়্যার আর্কিটেকচারকে বিশ্লেষণ করতে হবে।

আপনার ডিভাইসটি আসলে একটি একক কম্পিউটার নয়। একই চ্যাসিস বা আবরণের ভেতর দুটি সম্পূর্ণ আলাদা কম্পিউটার অবস্থান করে। এর প্রথম কম্পিউটারটি হলো Application Processor (AP)। এটি মূলত সেই CPU যা GrapheneOS রান করে, আপনার ডিসপ্লে রেন্ডার করে এবং আপনার এনক্রিপ্টেড মেসেজগুলো প্রসেস করে।

দ্বিতীয় কম্পিউটারটি হলো Cellular Baseband Processor (যাকে সংক্ষেপে শুধু "baseband" বলা হয়)। এই বেসব্যান্ড হলো একটি ডেডিকেটেড, অত্যন্ত জটিল মাইক্রো-কম্পিউটার, যার একমাত্র দায়িত্ব হলো সেলুলার টাওয়ারের সাথে আপনাকে সংযুক্ত করা রেডিও ওয়েভগুলোকে ম্যানেজ করা। সবচেয়ে গুরুত্বপূর্ণ বিষয় হলো, এই বেসব্যান্ডে Android রান করে না। এটি একটি অস্পষ্ট, মালিকানাধীন (proprietary), এবং ক্লোজড-সোর্স Real-Time Operating System (RTOS) ব্যবহার করে যা সাধারণত মডেম প্রস্তুতকারকদের (যেমন Qualcomm) দ্বারা ডেভেলপ করা হয়।

ঐতিহাসিকভাবেই বেসব্যান্ড ফার্মওয়্যার মারাত্মক মাত্রায় অনিরাপদ হিসেবে কুখ্যাত। লিগ্যাসি (legacy) সেলুলার প্রোটোকল সাপোর্ট করার জন্য এতে কয়েক দশক পুরোনো এমন মিলিয়ন মিলিয়ন লাইনের সেকেলে কোড রয়েছে। আর যেহেতু বেসব্যান্ড সরাসরি সেল টাওয়ারের সাথে বাতাসের মাধ্যমে (over the air) যোগাযোগ করে, তাই এটি অত্যন্ত ঝুঁকিপূর্ণ এবং বিশাল একটি রিমোট অ্যাটাক সারফেস তৈরি করে।

The IMSI Catcher Threat (Stingrays)

আইনশৃঙ্খলা রক্ষাকারী বাহিনী, গোয়েন্দা সংস্থা এবং অত্যাধুনিক সাইবার অপরাধী চক্রগুলো এই বেসব্যান্ড আর্কিটেকচারের দুর্বলতাকে কাজে লাগানোর জন্য এমন একটি ডিভাইস ব্যবহার করে, যা **IMSI Catcher** নামে পরিচিত (কথ্য ভাষায় এর ব্র্যান্ড নাম "Stingray" হিসেবে বেশি পরিচিত)।

একটি IMSI Catcher মূলত একটি ভুয়া, পোর্টেবল সেল টাওয়ার। এটি সাধারণত কোনো সার্ভেইল্যান্স ভ্যান (surveillance van), ড্রোন বা ব্যাগে করে জনবহুল পরিবেশে নিয়ে যাওয়া হয়। এই Stingray এমন একটি সিগন্যাল ব্রডকাস্ট করে যা আশেপাশের বৈধ সেল টাওয়ারগুলোর সিগন্যালের চেয়ে কৃত্রিমভাবে অনেক বেশি শক্তিশালী হয়।

সেলুলার বেসব্যান্ডগুলোকে এমনভাবে অ্যাগ্রেসিভ প্রোগ্রামিং করা হয়, যাতে তারা ব্যাটারি লাইফ বাঁচানোর জন্য সব সময় সম্ভাব্য সবচেয়ে শক্তিশালী সিগন্যালটি খুঁজে নেয়। ফলস্বরূপ, যখন আপনি কোনো Stingray-এর কাছাকাছি হেঁটে যান, তখন আপনার ফোনের বেসব্যান্ড অন্ধের মতো বৈধ নেটওয়ার্ক ত্যাগ করে সরাসরি সেই ভুয়া এবং ক্ষতিকর টাওয়ারটির সাথে কানেক্টেড হয়ে যায়।

একবার কানেক্টেড হওয়ার পর, Stingray নিজেকে চূড়ান্ত পর্যায়ের Man-in-the-Middle হিসেবে প্রতিষ্ঠিত করে ফেলে।

The Downgrade Attack

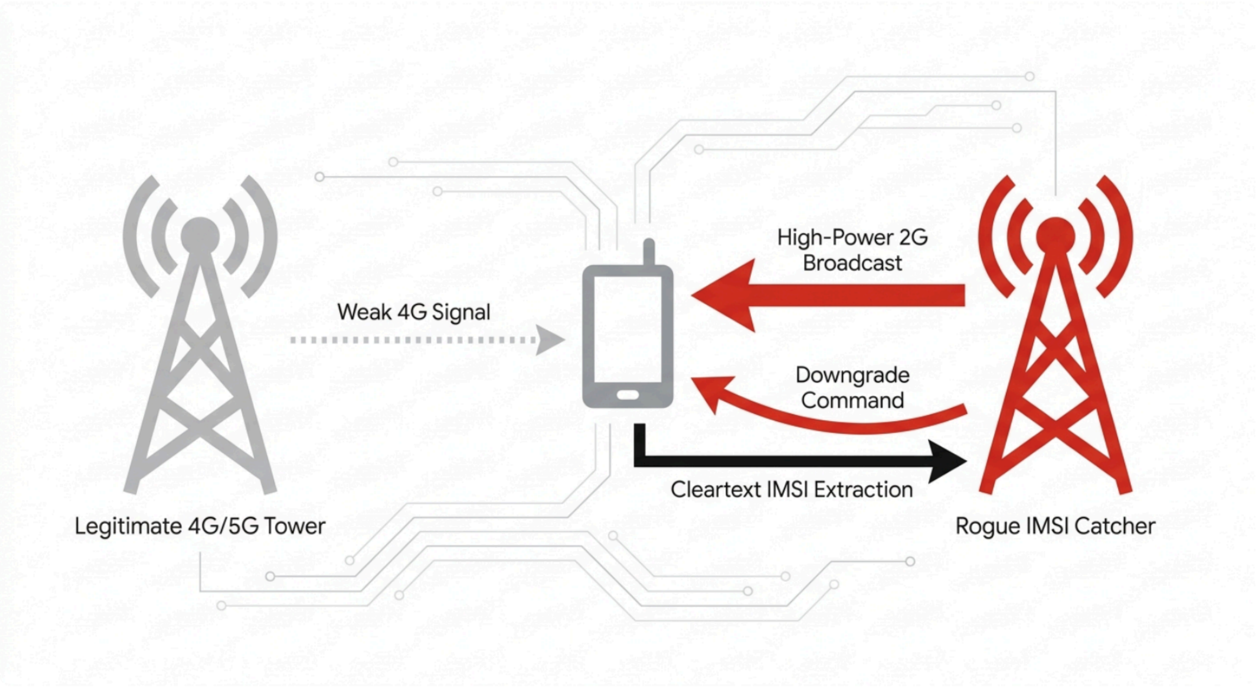
আধুনিক সেলুলার প্রোটোকলগুলো, যেমন 4G (LTE) এবং 5G-তে 'মিউচুয়াল অথেনটিকেশন' (mutual authentication) ব্যবস্থা থাকে। এর মানে হলো, নেটওয়ার্ক যেমন ফোনের আইডেন্টিটি ভেরিফাই করবে, তেমনি ফোনটিও নেটওয়ার্কের আইডেন্টিটি ভেরিফাই করবে। যেহেতু একটি Stingray-এর কাছে বৈধ টেলিকম প্রোভাইডারের ক্রিপ্টোগ্রাফিক কি (cryptographic keys) থাকে না, তাই এটি কোনোভাবেই একটি 4G কানেকশন সফলভাবে অথেনটিকেট করতে পারে না।

এই নিরাপত্তাব্যবস্থাকে বাইপাস করার জন্য Stingray তখন একটি **Downgrade Attack** বা ডাউনগ্রেড অ্যাটাক চালায়।

Stingray সক্রিয়ভাবে লোকাল 4G এবং 5G ফ্রিকোয়েন্সিগুলোকে জ্যাম (jam) করে দেয়, যার ফলে আধুনিক প্রোটোকলগুলোর জন্য একটি ডেড জোন (dead zone) তৈরি হয়। একই সময়ে এটি অত্যন্ত শক্তিশালী একটি 2G বা 3G সিগন্যাল ব্রডকাস্ট করতে থাকে।

4G-এর মতো পুরোনো 2G প্রোটোকলে 'মিউচুয়াল অথেনটিকেশন'-এর কোনো প্রয়োজন হয় *না*। ১৯৯০-এর দশকে যখন 2G ডিজাইন করা হয়েছিল, তখন ইঞ্জিনিয়াররা এমন পোর্টেবল, ভুয়া সেল টাওয়ারের অস্তিত্ব কল্পনাও করতে পারেননি। একটি 2G নেটওয়ার্কে ফোনকে টাওয়ারের কাছে নিজের আইডেন্টিটি প্রমাণ করতে হয়, কিন্তু টাওয়ারকে কখনো ফোনের কাছে তার আইডেন্টিটি প্রমাণ করতে হয় না। সবচেয়ে ভয়াবহ ব্যাপার হলো, 2G অত্যন্ত দুর্বল এনক্রিপশন ব্যবহার করে যা রিয়েল-টাইমেই ক্র্যাক করা সম্ভব।

Mechanics of Baseband Protocol Downgrade Attacks



Active cell-site simulators overpower legitimate infrastructure to force target devices into legacy network environments, exploiting the absence of mutual authentication to harvest permanent identifiers.

যখন আপনার সাধারণ কমার্শিয়াল স্মার্টফোনটি জ্যামিংয়ের কারণে 4G সিগন্যাল হারিয়ে ফেলে, তখন এটি তাৎক্ষণিকভাবে প্যানিক করে এবং স্বয়ংক্রিয়ভাবে তার কানেকশনকে Stingray-এর 2G নেটওয়ার্কে ডাউনগ্রেড করে ফেলে।

আর এই ডাউনগ্রেড হওয়ার মুহূর্তেই প্রতিপক্ষের জয় নিশ্চিত হয়। Stingray তখন ইনকামিং এবং আউটগোয়িং সমস্ত SMS টেক্সট মেসেজ ইন্টারসেপ্ট (intercept) করতে পারে, আনএনক্রিপ্টেড ফোন কলগুলো নীরবে রেকর্ড করতে পারে এবং আপনার ডিভাইসের সুনির্দিষ্ট ফিজিক্যাল লোকেশন ট্র্যাক করার জন্য এর ইউনিক হার্ডওয়্যার আইডেন্টিফায়ার (IMSI নম্বর) বের করে নিতে পারে।

The GrapheneOS Defense: LTE-Only Mode

GrapheneOS অত্যন্ত কঠোর এবং আপসহীন একটি বেসব্যান্ড কনফিগারেশন ডিপ্লয়মেন্টের মাধ্যমে Stingray-এর মতো থ্রেটগুলোকে পুরোপুরি নিষ্ক্রিয় করে দেয়: আর সেটি হলো **LTE-Only Mode**।

GrapheneOS এর অপারেটরকে সেলুলার কমিউনিকেশনের প্যারামিটারগুলো কঠোরভাবে নির্ধারণ করার সুযোগ দেয়। একজন এলিট অপারেটর **Settings > Network & internet > SIMs > Preferred network type**-এ গিয়ে সুনির্দিষ্টভাবে **LTE Only** নির্বাচন করবেন।

এই একটি মাত্র কনফিগারেশন পরিবর্তন বেসব্যান্ড প্রসেসরের আচরণকে মৌলিকভাবে বদলে দেয়। এর মাধ্যমে আপনি গাণিতিকভাবে বেসব্যান্ডকে পুরোনো 2G বা 3G কোড লোড করতে, এক্সিকিউট করতে বা তার মাধ্যমে কোনো ধরনের যোগাযোগ স্থাপন করতে সম্পূর্ণ নিষেধ করে দিচ্ছেন।

আপনি যদি কোনো সক্রিয় Stingray-এর রেডিয়াসের ভেতর প্রবেশ করেন, তবে প্রতিপক্ষ আপনার 4G সিগন্যাল জ্যাম করে দেবে। কিন্তু যেহেতু আপনি LTE-Only মোড এনফোর্স (enforce) করেছেন, তাই ডাউনগ্রেড অ্যাটাক করা তখন অসম্ভব হয়ে দাঁড়াবে। আপনার বেসব্যান্ড তখন কোনোভাবেই সেই ক্ষতিকর 2G নেটওয়ার্কে ফলব্যাক (fallback) করতে পারবে না।

প্রতিপক্ষের কাছে নিজের কমিউনিকেশন সারেন্ডার করার বদলে, আপনার ডিভাইসটি স্রেফ "No Service" রিপোর্ট করবে এবং সম্পূর্ণ সেলুলার গ্রিড থেকে বিচ্ছিন্ন হয়ে যাবে। একটি উচ্চ-নিরাপত্তা পরিবেশে, কম্প্রোমাইজড বা অনিরাপদ যোগাযোগের চেয়ে এই নীরবতা বহুগুণ বেশি নিরাপদ।

> ***OpSec Note on 5G:*** *যদিও 4G-এর মতো 5G-ও মিউচুয়াল অথেনটিকেশন প্রদান করে, তবে 5G প্রোটোকলটি অবিশ্বাস্যভাবে জটিল ("bleeding-edge code")। সাইবার সিকিউরিটির জগতে, জটিলতা হলো নিরাপত্তার শত্রু। বিপুল পরিমাণের নতুন এবং আন-অডিটেড (un-audited) কোড বেসব্যান্ডে অজানা জিরো-ডে (zero-day) দুর্বলতা থাকার সম্ভাবনাকে মারাত্মকভাবে বাড়িয়ে দেয়। আপনার ডিভাইসটিকে শুধুমাত্র LTE (4G)-এর মধ্যে সীমাবদ্ধ রাখার মাধ্যমে, আপনি একই সাথে 2G/3G-এর প্রাচীন দুর্বলতাগুলো নিষ্ক্রিয় করছেন এবং 5G-এর মতো ব্লিডিং-এজ (bleeding-edge) অ্যাটাক সারফেসকেও সম্পূর্ণ নির্মূল করছেন।*

The Ultimate Fail-Safe: Baseband Hardware Isolation

আমরা সফলভাবে ডাউনগ্রেড অ্যাটাক প্রতিহত করেছি, কিন্তু আমাদের সবচেয়ে খারাপ পরিস্থিতির (worst-case scenario) জন্যও প্রস্তুত থাকতে হবে। কী হবে যদি কোনো অত্যাধুনিক, রাষ্ট্র-সমর্থিত প্রতিপক্ষ খোদ LTE প্রোটোকলের জন্যই একটি রিমোট জিরো-ডে (zero-day) এক্সপ্লয়েট ডেভেলপ করে ফেলে? কী হবে যদি তারা সেল টাওয়ারকে পুরোপুরি বাইপাস করে,

4G-এর মাধ্যমে সরাসরি আপনার ডিভাইসে একটি ওয়েপোনাইজড রেডিও প্যাকেট (weaponized radio packet) পাঠায় এবং সফলভাবে Baseband Processor-কে কম্প্রোমাইজ বা হ্যাক করে ফেলে?

২০১০-এর দশকের শুরুর দিকে বেসব্যান্ড কম্প্রোমাইজড হওয়ার অর্থ ছিল পুরো ডিভাইসের নিয়ন্ত্রণ হারিয়ে ফেলা। কারণ, শুরুর দিকের স্মার্টফোনগুলোতে Application Processor-এর RAM-এ Baseband Processor-এর Direct Memory Access (DMA) ছিল। এর ফলে, কোনো আক্রমণকারী যদি রেডিও ওয়েভ ব্যবহার করে বেসব্যান্ড হ্যাক করতে পারত, তবে সে অনায়াসেই মূল সিস্টেম মেমোরি রিড করতে পারত, যার মাধ্যমে আপনার এনক্রিপশন কি (encryption keys) চুরি করা এবং আপনার ব্যক্তিগত মেসেজ টাইপ করার সময়ই সেগুলো পড়া সম্ভব ছিল।

GrapheneOS এবং আধুনিক Pixel হার্ডওয়্যারগুলো **IOMMU Isolation**-এর মাধ্যমে এই ধ্বংসাত্মক দুর্বলতাটিকে পুরোপুরি নির্মূল করেছে।

Input-Output Memory Management Unit (IOMMU) হলো একটি ফিজিক্যাল হার্ডওয়্যার ফায়ারওয়াল, যা সরাসরি ডিভাইসের সিলিকন বা মূল চিপের ভেতরেই তৈরি করা থাকে। এটি মূলত Baseband Processor এবং Application Processor-এর ঠিক মাঝখানে অবস্থান করে।

IOMMU নিশ্চিত করে যে, Baseband Processor-কে অত্যন্ত কঠোরভাবে সীমাবদ্ধ রাখা হয়েছে। যেকোনোভাবে মূল সিস্টেম মেমোরি রিড বা রাইট করা এর জন্য ফিজিক্যালি নিষিদ্ধ। বেসব্যান্ডকে শুধুমাত্র একটি ছোট, কঠোরভাবে মনিটরকৃত এবং গাণিতিকভাবে সংজ্ঞায়িত বাফার জোনে (buffer zone) অ্যাক্সেস করার অনুমতি দেওয়া হয়, যা শুধুমাত্র নেটওয়ার্ক প্যাকেট আদান-প্রদানের কাজেই কঠোরভাবে ব্যবহৃত হয়।

এমনকি কোনো রাষ্ট্র-সমর্থিত হ্যাকার গোষ্ঠীও যদি আপনার সেলুলার রেডিওর বিরুদ্ধে কয়েক মিলিয়ন ডলার মূল্যের কোনো জিরো-ডে (zero-day) এক্সপ্লয়েট সফলভাবে লঞ্চ করে এবং Baseband Processor-এর ওপর সম্পূর্ণ নিয়ন্ত্রণ নিয়ে নেয়, তবুও আক্রমণটি ঠিক সেখানেই শেষ হয়ে যায়। ম্যালওয়্যারটি ফিজিক্যালি বেসব্যান্ড সিলিকনের ভেতরেই আটকা পড়ে থাকে। এটি কোনোভাবেই IOMMU ফায়ারওয়াল অতিক্রম করতে পারে না। এটি GrapheneOS-এ অ্যাক্সেস করতে পারে না, আপনার এনক্রিপ্টেড Signal মেসেজগুলো পড়তে পারে না, আপনার ছবি দেখতে পারে না এবং আপনার পাসওয়ার্ডগুলোও চুরি করতে পারে না।

Stingray ডাউনগ্রেড অ্যাটাক ঠেকাতে LTE-Only মোড বাধ্যতামূলক করে এবং বেসব্যান্ড জিরো-ডে (zero-day) ম্যালওয়্যারকে কোয়ারেন্টাইন করতে IOMMU হার্ডওয়্যার আইসোলেশনের ওপর নির্ভর করার মাধ্যমে, GrapheneOS এটি নিশ্চিত করে যে আপনার চারপাশে ঘিরে থাকা অদৃশ্য রেডিও ওয়েভগুলো কোনোভাবেই এই নিরাপত্তা দুর্গের দেয়াল ভেদ করতে পারবে না।

২.১৪ Phantom Comms: Virtual eSIMs এবং VoIP

২.১৩ নং সেকশনে, আমরা ভুয়া সেল টাওয়ারের বিরুদ্ধে ডিভাইসের ফিজিক্যাল বেসব্যান্ড প্রসেসরকে হার্ডেনড (hardened) বা সুরক্ষিত করেছি। তবে, আপনি যদি একটি বৈধ, ক্রিপ্টোগ্রাফিক্যালি ভেরিফায়েড সেলুলার টাওয়ারের সাথেও কানেক্টেড থাকেন, তবুও আপনি একটি বিশাল কাঠামোগত দুর্বলতার (structural vulnerability) সম্মুখীন হবেন: আর তা হলো স্বয়ং টেলিকম ক্যারিয়ার বা অপারেটরটি।

যখন আপনি একটি ট্র্যাডিশনাল বা গতানুগতিক SIM কার্ড কেনেন এবং কোনো বড় টেলিকম প্রোভাইডারের (যেমন AT&T, Verizon, অথবা Vodafone) সাথে চুক্তিবদ্ধ হন, তখন আপনি মূলত একটি নজরদারি অংশীদারিত্বে (surveillance partnership) প্রবেশ করেন। ক্যারিয়ার আপনার কাছে আপনার সরকারি আইডি, বাড়ির ঠিকানা এবং বিলিং ইনফরমেশন দাবি করে। এর বিনিময়ে তারা আপনাকে একটি সাধারণ ফোন নম্বর এবং তাদের নেটওয়ার্কে অ্যাক্সেস দেয়।

সেই মুহূর্ত থেকে, টেলিকম ক্যারিয়ার আপনার সমস্ত কমিউনিকেশন মেটাডেটা (communication metadata)-র ওপর সম্পূর্ণ নজরদারির সক্ষমতা পেয়ে যায়। আপনি কাকে কল করছেন, কতক্ষণ কথা বলছেন এবং কাকে কী SMS টেক্সট মেসেজ পাঠাচ্ছেন—তার সবকিছুই তারা লগ (log) করে রাখে। শুধু তাই নয়, যেহেতু সিগন্যাল ধরে রাখার জন্য আপনার ডিভাইসটিকে প্রতিনিয়ত তাদের টাওয়ারে পিং (ping) করতে হয়, তাই ক্যারিয়ার সেল টাওয়ার ট্রায়াঙ্গুলেশন (cell tower triangulation)-এর মাধ্যমে রিয়েল-টাইমে আপনার ফিজিক্যাল লোকেশনের একটি নিরবচ্ছিন্ন ম্যাপ তৈরি করতে পারে—যেই ডেটা ঐতিহাসিকভাবেই তারা কোনো ওয়ারেন্ট ছাড়াই থার্ড-পার্টি ডেটা ব্রোকার এবং আইনশৃঙ্খলা রক্ষাকারী বাহিনীর কাছে বিক্রি করে আসছে।

এলিট অপারেশনাল সিকিউরিটি (operational security) বা সর্বোচ্চ নিরাপত্তা অর্জন করতে হলে আমাদের অবশ্যই এই সম্পর্কের সম্পূর্ণ অবসান ঘটাতে হবে। আমাদের ট্র্যাডিশনাল ফোন নম্বরের ধারণাটিকেই পুরোপুরি বর্জন করতে হবে।

The Triad of Carrier Vulnerabilities

একটি ট্র্যাডিশনাল টেলিকম ক্যারিয়ারের ওপর নির্ভর করলে আপনি তিনটি জটিল এবং গুরুতর দুর্বলতার সম্মুখীন হবেন:

১. SS7 Interception: Signaling System 7 (SS7) হলো এমন একটি গ্লোবাল রাউটিং প্রোটোকল যা বিভিন্ন টেলিকম ক্যারিয়ারকে একে অপরের সাথে যোগাযোগ করার সুযোগ দেয় (যেমন, লন্ডন থেকে করা একটি কল কীভাবে টোকিওর একটি ফোনে সফলভাবে পৌঁছায়)। SS7 প্রোটোকলটি ১৯৭৫ সালে ডিজাইন করা হয়েছিল এবং এতে ফান্ডামেন্টাল অথেনটিকেশনের (fundamental authentication) অভাব রয়েছে। অত্যাধুনিক সাইবার অপরাধী এবং রাষ্ট্র-সমর্থিত হ্যাকাররা (state

actors) নিয়মিতভাবেই SS7 এক্সপ্লয়েট করার মাধ্যমে আপনার ফিজিক্যাল ডিভাইস স্পর্শ না করে বা কোনো ম্যালওয়্যার ইনস্টল না করেই বিশ্বের যেকোনো প্রান্ত থেকে আপনার SMS মেসেজ ইন্টারসেপ্ট (intercept) করতে এবং ভয়েস কল রেকর্ড করতে পারে।

২. SIM Swapping: ২.৯ নং সেকশনে যেমনটি আলোচনা করা হয়েছে, ট্র্যাডিশনাল ফোন নম্বরগুলো সোশ্যাল ইঞ্জিনিয়ারিংয়ের (social engineering) কাছে অত্যন্ত দুর্বল। একজন প্রতিপক্ষ সহজেই কোনো নিম্নপদস্থ টেলিকম কর্মীকে ঘুষ দিয়ে বা প্রতারণা করে আপনার ফোন নম্বরটি একটি নতুন SIM কার্ডে ট্রান্সফার করে নিতে পারে। যদি আপনার ব্যাংক বা ইমেইল অ্যাকাউন্ট টু-ফ্যাক্টর অথেনটিকেশনের (two-factor authentication) জন্য SMS ব্যবহার করে, তবে আক্রমণকারী তাৎক্ষণিকভাবে আপনার নিরাপত্তাব্যবস্থাকে বাইপাস করে ফেলবে।

৩. The Metadata Dragnet: আপনি যদি এনক্রিপ্টেড মেসেজিং অ্যাপসও ব্যবহার করেন, তবুও ক্যারিয়ার আপনার লিগ্যাসি (legacy) সেলুলার কলগুলোর মেটাডেটা ঠিকই দেখতে পায়। তারা জানে যে আপনি দুপুর ২:০০ টায় ঠিক ১৪ মিনিটের জন্য একটি নির্দিষ্ট মেডিকেল ক্লিনিকে কল করেছিলেন। ইন্টেলিজেন্স অ্যানালাইসিসে (intelligence analysis), মূল কথোপকথনের বিষয়বস্তুর চেয়েও মেটাডেটা অনেক সময় বেশি তথ্য প্রকাশ করে দেয়।

The Phantom Strategy: Data-Only eSIMs

টেলিকম একচেটিয়া আধিপত্যের (telecom monopoly) একমাত্র সমাধান হলো **Data-Only eSIM** কৌশল বা স্ট্র্যাটেজি।

আধুনিক GrapheneOS ডিভাইসগুলো (Pixel 4 এবং এর পরবর্তী মডেল) এমবেডেড সিম বা eSIM (embedded SIMs) সাপোর্ট করে। একটি eSIM হলো মাদারবোর্ডের ভেতরে সরাসরি বিল্ট-ইন থাকা একটি প্রোগ্রামেবল মাইক্রোচিপ (programmable microchip), যা কোনো ফিজিক্যাল প্লাস্টিক কার্ডের প্রয়োজন ছাড়াই একটি QR কোড স্ক্যান করে ডিজিটালভাবে সেলুলার প্ল্যান ডাউনলোড করার সুবিধা দেয়।

একজন এলিট অপারেটর ট্র্যাডিশনাল ফোন প্ল্যানের জন্য কোনো চুক্তি সই করার বদলে একটি প্রিপেইড, অ্যানোনিমাস এবং ভার্চুয়াল 'ডেটা-অনলি' (data-only) eSIM ব্যবহার করেন। *Silent.Link*, *Airalo*, বা বিভিন্ন গ্লোবাল ট্রাভেল eSIM ভেন্ডররা এমন প্ল্যান বিক্রি করে যা শুধুমাত্র র (raw) ইন্টারনেট ব্যান্ডউইথ প্রদান করে আর কিচ্ছু না।

The Tactical Advantages:

- **Zero Legacy Capability:** একটি ডেটা-অনলি eSIM-এ কোনো ট্র্যাডিশনাল ফোন নম্বর দেওয়া হয় না। এর মাধ্যমে আপনি আনএনক্রিপ্টেড (unencrypted) সেলুলার ফোন কল করতে পারবেন না, এমনকি কোনো আনএনক্রিপ্টেড SMS টেক্সট মেসেজও পাঠাতে বা রিসিভ করতে পারবেন না। গাণিতিকভাবে এই সক্ষমতাগুলোকে

বাদ দেওয়ার মাধ্যমে, আপনি আপনার SS7 অ্যাটাক সারফেস (attack surface) এবং SIM Swapping-এর দুর্বলতাকে নামিয়ে একেবারে শূন্যে (zero) নিয়ে আসতে পারেন।

- **Anonymity:** অনেক ভার্চুয়াল eSIM প্রোভাইডারেরই Know Your Customer (KYC) বা কোনো আইডেন্টিটি ভেরিফিকেশনের প্রয়োজন হয় না। একটি VPN-এ কানেক্টেড থাকা অবস্থায় আপনি ক্রিপ্টোকারেন্সি (যেমন Monero) অথবা একটি মাস্কড (masked) ক্রেডিট কার্ড ব্যবহার করে অনায়াসেই এই ডেটা প্ল্যানটি কিনতে পারেন। সেলুলার নেটওয়ার্ক আপনাকে ডেটা প্রদান করবে ঠিকই, কিন্তু আপনি আসলে কে তা তাদের জানার কোনো উপায় থাকবে না।
- **Burner Flexibility:** আপনি যদি সন্দেহ করেন যে আপনার IMSI নম্বরটি কোনো শত্রু পক্ষ ট্র্যাক করছে, তবে SIM পরিবর্তন করার জন্য আপনার কোনো টেলিকম স্টোরে যাওয়ার দরকার নেই। আপনি শুধু GrapheneOS সেটিংস থেকে আপনার আগের eSIM প্রোফাইলটি ডিলিট করবেন, একটি নতুন QR কোড স্ক্যান করবেন এবং সেকেন্ডের মধ্যেই সম্পূর্ণ নতুন ও আনট্রেসেবল (untraceable) একটি সেলুলার আইডেন্টিটি বা পরিচয় তৈরি করে ফেলতে পারবেন।

Cryptographic VoIP Routing

যদি আপনার কোনো ফোন নম্বর না থাকে এবং আপনি কোনো ট্র্যাডিশনাল কল করতে না পারেন, তবে আপনি কীভাবে যোগাযোগ করবেন?

এর সমাধান হলো, আপনি আপনার যোগাযোগের ১০০% ইন্টারনেটের মাধ্যমে **End-to-End Encrypted Voice over IP (VoIP)** অ্যাপ্লিকেশন দিয়ে রাউট (route) করবেন।

যেহেতু আপনার ডিভাইসটি শুধুমাত্র র (raw) ইন্টারনেট ডেটার জন্য সেলুলার টাওয়ারের সাথে সংযুক্ত থাকে, তাই এক্ষেত্রে টেলিকম ক্যারিয়ার কেবল একটি 'ডাম্ব পাইপ' (dumb pipe) বা নিছক মাধ্যম হিসেবে কাজ করে। আপনি টাওয়ারের সাথে কানেক্ট করার পরপরই তাৎক্ষণিকভাবে আপনার WireGuard VPN টানেলটি এস্টাবলিশ করবেন (যেভাবে ২.১১ নং সেকশনে বিস্তারিত বলা হয়েছে) এবং এই টানেলের ভেতর থেকেই আপনার সমস্ত যোগাযোগ পরিচালনা করবেন।

১. Signal (For Known Contacts): পরিচিত কন্টাক্টদের সাথে যোগাযোগের জন্য Signal ক্রিস্টাল-ক্লিয়ার এবং এনক্রিপ্টেড ভয়েস ও ভিডিও কলের পাশাপাশি এনক্রিপ্টেড মেসেজিং সুবিধা প্রদান করে। যেহেতু এটি আপনার ডেটা কানেকশনের ওপর দিয়ে রাউটিং হয়, তাই টেলিকম ক্যারিয়ার কোনো অডিও ইন্টারসেপ্ট করতে পারে না এবং আপনি কার সাথে কথা বলছেন সে সম্পর্কেও তাদের কোনো ধারণা থাকে না। *(নোট: অ্যাকাউন্ট রেজিস্ট্রেশনের জন্য Signal-এ প্রাথমিকভাবে একটি ফোন নম্বরের প্রয়োজন হয়। এলিট অপারেটররা শুধুমাত্র প্রাথমিক রেজিস্ট্রেশনের টেক্সট মেসেজটি পাওয়ার জন্য একটি সস্তা, অস্থায়ী বার্নার (burner) VoIP নম্বর কেনেন অথবা MySudo-এর মতো একটি সুরক্ষিত সার্ভিস ব্যবহার করেন। এরপর পরবর্তী সমস্ত যোগাযোগের জন্য তারা ডেটা-অনলি eSIM-এর ওপর নির্ভর করেন)।*

২. Session (For Absolute Anonymity): যেসব যোগাযোগের ক্ষেত্রে সম্পূর্ণ অ্যানোনিমিটি বা পরিচয়ের পরম গোপনীয়তা প্রয়োজন, সেসব ক্ষেত্রে অপারেটররা **Session** অ্যাপ্লিকেশন ব্যবহার করেন। Session হলো একটি ডিসেন্ট্রালাইজড (decentralized), এন্ড-টু-এন্ড এনক্রিপ্টেড মেসেঞ্জার, যেখানে রেজিস্টার করার জন্য কোনো ফোন নম্বরের প্রয়োজন হয় না। এটি একটি অনিয়ন-রাউটেড নেটওয়ার্ক (onion-routed network) ব্যবহার করে মেসেজগুলো রাউট করে (অনেকটা Tor-এর মতো)। আপনি শুধু একটি Session ID জেনারেট করবেন এবং ডেটা কানেকশনের ওপর দিয়ে অত্যন্ত সুরক্ষিতভাবে যোগাযোগ করতে পারবেন।

Neutralizing the Carrier

এই Phantom Comms স্ট্র্যাটেজিটি প্রয়োগ করার মাধ্যমে আপনি টেলিকম ইনফ্রাস্ট্রাকচার বা অবকাঠামোটিকে পুরোপুরি অন্ধ (blind) করে দেন।

আপনি যখন আপনার ডেটা-অনলি eSIM-এর মাধ্যমে Signal ব্যবহার করে আপনার কোনো সহকর্মীকে একটি VoIP কল করেন, তখন লোকাল সেলুলার টাওয়ার আপনার সিগন্যাল প্রসেস করে ঠিকই, কিন্তু সেখানে যে কোনো ফোন কল চলছে সে সম্পর্কে তারা সম্পূর্ণ অজ্ঞ থাকে। টেলিকম ক্যারিয়ারের নজরদারি যন্ত্রপাতি আপনার কানেকশনের দিকে তাকিয়ে শুধুমাত্র একটি অগোছালো ও এনক্রিপ্টেড ডেটা স্ট্রিম (stream of encrypted data) দেখতে পায়, যা একটি VPN সার্ভারের দিকে প্রবাহিত হচ্ছে। তারা কোনো SS7 ডাউনগ্রেড অ্যাটাক চালাতে পারে না, কারণ ডাউনগ্রেড করার মতো কোনো সেলুলার ভয়েস ক্যাপাবিলিটিই আপনার নেই। তারা কোনো SIM Swap অ্যাটাকও চালাতে পারে না, কারণ সেই SIM-এর সাথে চুরি করার মতো কোনো ফোন নম্বর যুক্তই নেই।

আপনি সফলভাবে আপনার যোগাযোগ ব্যবস্থাকে বিংশ শতাব্দীর সেই সেকেলে এবং অনিরাপদ ইনফ্রাস্ট্রাকচার থেকে ডিকাপল (decoupled) বা বিচ্ছিন্ন করে ফেলেছেন। আপনি এখন আর কোনো ফোন কল করছেন না; আপনি মূলত ক্রিপ্টোগ্রাফিক ডেটা প্যাকেট ট্রান্সমিট করছেন। এই ফ্যান্টম স্ট্র্যাটেজিটি নিশ্চিত করে যে আপনার ভয়েস, আপনার মেসেজ এবং আপনার মেটাডেটা একান্তই আপনার নিজস্ব সম্পত্তি।

২.১৫ Travel Security: Understanding BFU vs AFU

এতক্ষণ আমরা যেসব ডিজিটাল ডিফেন্স তৈরি করেছি, সেগুলো রিমোট অ্যাটাক (যেমন- ম্যালওয়্যার, নেটওয়ার্ক স্নিফিং এবং ক্ষতিকর বেসব্যান্ড) প্রতিহত করার জন্য ডিজাইন করা। কিন্তু এমন একটি দৃশ্যকল্প (scenario) রয়েছে যেখানে আপনার প্রতিপক্ষের কোনো নেটওয়ার্ক ফায়ারওয়াল বাইপাস করার প্রয়োজন নেই, কারণ আপনার ডিভাইসটি তখন সরাসরি তাদের হাতেই থাকে।

ডিজিটাল প্রাইভেসি বা গোপনীয়তার ক্ষেত্রে আন্তর্জাতিক সীমান্ত বা বর্ডার ক্রসিংগুলো বিশ্বের সবচেয়ে প্রতিকূল ফিজিক্যাল এনভায়রনমেন্টগুলোর একটি। যখন আপনি কোনো বর্ডার ক্রস করেন, তখন সার্চ এবং সিজার (search and seizure) সংক্রান্ত সাধারণ সাংবিধানিক ও আইনি সুরক্ষাগুলো প্রায়শই স্থগিত থাকে। অনেক এখতিয়ারে (jurisdiction) বর্ডার এজেন্টদের কাছে এমন অবারিত আইনি ক্ষমতা থাকে যার মাধ্যমে তারা আপনাকে আটকে রাখতে পারে, আপনার ইলেক্ট্রনিক্স বাজেয়াপ্ত করতে পারে এবং ফিজিক্যালি সেগুলোকে অত্যাধুনিক ফরেনসিক এক্সট্রাকশন মেশিনের (যেমন Cellebrite বা GrayKey) সাথে যুক্ত করতে পারে।

যদি কোনো বর্ডার এজেন্ট আপনাকে আপনার ডিভাইসটি সারেন্ডার করতে বা তাদের হাতে তুলে দিতে বলে, তবে সেই ইন্টার‍্যাকশনের ঠিক আগের কয়েক সেকেন্ডে আপনার ফিজিক্যাল আচরণই নির্ধারণ করে দেবে যে আপনার ইন্টেলিজেন্স বা ডেটা সুরক্ষিত থাকবে, নাকি পুরোপুরি কম্প্রোমাইজড হয়ে যাবে।

প্রপার ট্রাভেল সিকিউরিটি বা ভ্রমণের সময় যথাযথ নিরাপত্তা নিশ্চিত করতে হলে আপনাকে অবশ্যই আধুনিক ক্রিপ্টোগ্রাফির একটি চরম দুর্বলতা বুঝতে হবে: আর তা হলো লক স্ক্রিনের মায়া বা বিভ্রম (illusion of the lock screen)।

The Illusion of the Lock Screen

আপনি যখন কোনো ইমেইল পড়া শেষ করেন, তখন ডিভাইসের পাশে থাকা ফিজিক্যাল পাওয়ার বাটনে চাপ দেন। স্ক্রিনটি অন্ধকার হয়ে যায়। এরপর যদি কেউ আপনার ডিভাইসটি হাতে নেয়, তবে আপনার PIN বা ফিঙ্গারপ্রিন্ট জানা ছাড়া তারা আপনার ইমেইলগুলো পড়তে পারে না।

আপনি নিজেকে সুরক্ষিত মনে করেন। আপনার মনে হয় ডিভাইসটি লক করা আছে।

কিন্তু ক্রিপ্টোগ্রাফিক দৃষ্টিকোণ থেকে, এই অনুভূতিটি একটি ভয়ংকর বিভ্রম। কেন এটি বিভ্রম তা বুঝতে হলে, আমাদের Android-এর File-Based Encryption (FBE)-এর মেকানিজম বা কার্যপদ্ধতিটি বুঝতে হবে।

আপনি যখন প্রথমবার আপনার ডিভাইসটি চালু করেন, তখন আপনার ডেটা হার্ডড্রাইভে গাণিতিকভাবে সিল (sealed) বা তালাবদ্ধ অবস্থায় থাকে। কিন্তু প্রথমবার আপনার পাসওয়ার্ড টাইপ করার সাথে সাথেই অপারেটিং সিস্টেম সেই পাসওয়ার্ড ব্যবহার করে মাস্টার ডিক্রিপশন কি (master decryption keys) তৈরি করে। এরপর এটি সেই ডিক্রিপশন কি-গুলোকে ডিভাইসের Random Access Memory (RAM)-এ লোড করে নেয়।

যতক্ষণ এই কি-গুলো RAM-এ থাকে, ততক্ষণ আপনার ফাইলগুলো অন-দ্য-ফ্লাই (on the fly) ডিক্রিপ্ট হতে থাকে যাতে আপনার অ্যাপ্লিকেশনগুলো সঠিকভাবে কাজ করতে পারে। আপনি যখন স্ক্রিন "লক" করার জন্য পাওয়ার বাটনে চাপ দেন, তখন অপারেটিং সিস্টেম RAM থেকে সেই কি-গুলোকে ডিলিট করে *না*। এটি স্রেফ ডিসপ্লের ওপর একটি গ্রাফিকাল লক স্ক্রিন বসিয়ে দেয়।

ফলে স্ক্রিন লক করা থাকলেও, ক্রিপ্টোগ্রাফিক কি-গুলো কিন্তু তখনও 'হট' (hot) বা সক্রিয় অবস্থাতেই থাকে।

The AFU State (After First Unlock)

ডিজিটাল ফরেনসিকে, কোনো ডিভাইসের 'হট' কি (hot keys) যখন RAM-এ সক্রিয় থাকে, তখন সেই ডিভাইসটিকে **AFU (After First Unlock)** স্টেটে বা অবস্থায় আছে বলে ধরা হয়।

বর্ডার এজেন্টরা যদি আপনার ডিভাইসটিকে AFU স্টেটে থাকা অবস্থায় বাজেয়াপ্ত করে, তবে ডেটা এক্সট্র্যাক্ট করার জন্য তাদের আপনার PIN-এর কোনো প্রয়োজন নেই। তারা ডিভাইসটিকে একটি ভেতরের রুমে নিয়ে যায়, USB-C পোর্টে একটি বিশেষ ফরেনসিক ক্যাবল প্লাগ করে এবং অত্যন্ত ক্লাসিফায়েড বা গোপনীয় এক্সপ্লয়েট (exploits) রান করে, যেগুলো গ্রাফিকাল লক স্ক্রিন বাইপাস করে সরাসরি RAM-এ অ্যাক্সেস নেওয়ার জন্য ডিজাইন করা।

যেহেতু ডিক্রিপশন কি-গুলো তখনও মেমোরিতে বসে আছে, ফরেনসিক মেশিনটি স্রেফ সেই কি-গুলোকে গ্র্যাব বা ক্যাপচার করে এবং সেগুলো ব্যবহার করে আপনার স্টোরেজ ডিক্রিপ্ট করে ফেলে। এক্সট্রাকশন টুলটি তখন আপনার হেভি বা শক্তিশালী এনক্রিপশনকে পুরোপুরি বাইপাস করে অনায়াসেই আপনার টেক্সট মেসেজ, ছবি, ওয়েব হিস্ট্রি এবং কন্টাক্ট লিস্ট বের করে আনতে পারে।

AFU স্টেটে থাকা কোনো ডিভাইস সারেন্ডার করা আর নিজের ঘরের দরজা লক করে তালার ভেতরে ফিজিক্যাল চাবিটি রেখে আসার মধ্যে কোনো পার্থক্য নেই।

> ⚠ ***Warning:*** *স্ক্রিন লক থাকার অর্থ এই নয় যে আপনার ডেটা নিরাপদ। আপনার ডিভাইসটি যদি সর্বশেষ রিবুটের পর অন্তত একবারও আনলক করা হয়ে থাকে, তবে এর ডিক্রিপশন কি-গুলো RAM-এ লাইভ বা সক্রিয় আছে। বর্ডার ক্রসিং, রাজনৈতিক বিক্ষোভ বা কোনো প্রতিকূল স্থাপনার মতো যেকোনো হাই-রিস্ক ফিজিক্যাল এনভায়রনমেন্টে প্রবেশের আগে সর্বদা আপনার ডিভাইসটি* ***reboot*** *করুন। একটি রিবুট হতে মাত্র তিন সেকেন্ড সময় লাগে, কিন্তু এটি মেমোরি থেকে সম্পূর্ণ কি-গুলোকে মুছে ফেলে (wipes)।*

The BFU State (Before First Unlock)

ফরেনসিক থ্রেট বা হুমকিটি নিষ্ক্রিয় করতে হলে, আপনাকে অবশ্যই ডিভাইসটিকে এর ঠিক বিপরীত ক্রিপ্টোগ্রাফিক অবস্থায় বা কন্ডিশনে নিতে বাধ্য করতে হবে: যাকে বলা হয় **BFU (Before First Unlock)** স্টেট।

একটি ডিভাইস পুরোপুরি শাটডাউন বা রিবুট হওয়ার ঠিক পর পরই, অর্থাৎ প্রথমবার PIN টাইপ করার *আগে* যে অবস্থায় থাকে, সেটিই হলো BFU স্টেট। এই স্টেটে, RAM পুরোপুরি খালি থাকে। কোনো ডিক্রিপশন কি জেনারেট বা তৈরি হয় না। স্টোরেজ ড্রাইভের সমস্ত ইউজার ডেটা তখন "বিশ্রামে" (at rest) এবং গাণিতিকভাবে সিল (sealed) করা থাকে।

বর্ডার এজেন্টরা যদি BFU স্টেটে থাকা একটি ডিভাইস বাজেয়াপ্ত করে এবং এটিকে কোনো Cellebrite মেশিনে প্লাগ করে, তবে ফরেনসিক সফটওয়্যারটি মূলত একটি অভেদ্য ক্রিপ্টোগ্রাফিক দেয়ালে গিয়ে ধাক্কা খায়। মেশিনটি কি-গুলোর জন্য RAM স্ক্যান করার চেষ্টা করে কিন্তু কিছুই খুঁজে পায় না। এটি হার্ডড্রাইভ রিড করার চেষ্টা করে কিন্তু এনক্রিপ্টেড এবং দুর্বোধ্য আবোলতাবোল তথ্যের (encrypted gibberish) এক অভেদ্য দেয়াল ছাড়া আর কিছুই রিসিভ করে না।

কি (keys) জেনারেট করার জন্য ফিজিক্যাল PIN ছাড়া ডিভাইসটি প্রতিপক্ষের কাছে সম্পূর্ণ অর্থহীন। এখান থেকে কোনোভাবেই ডেটা এক্সট্রাক্ট করা সম্ভব নয়।

The Tactical Protocol: The Reboot Rule

এই দুই স্টেটের মধ্যকার বিশাল পার্থক্যটি জানার মাধ্যমে আপনি ফিজিক্যাল OpSec-এর সবচেয়ে গুরুত্বপূর্ণ ট্যাকটিকাল প্রোটোকলটি প্রয়োগ করতে সক্ষম হবেন: **The Reboot Rule**।

যখনই আপনি কোনো হাই-রিস্ক বা উচ্চ-ঝুঁকিপূর্ণ ফিজিক্যাল এনভায়রনমেন্টের কাছাকাছি পৌঁছাচ্ছেন—তা সে কোনো আন্তর্জাতিক বিমানবন্দরের কাস্টমস ডেস্কে পা রাখা হোক, কোনো প্রতিকূল কর্পোরেট ফ্যাসিলিটিতে প্রবেশ করা হোক, কিংবা কোনো সংবেদনশীল রাজনৈতিক বিক্ষোভে অংশ নেওয়া হোক—আপনাকে অবশ্যই জোরপূর্বক আপনার ডিভাইসটিকে AFU থেকে BFU স্টেটে ট্রানজিশন (transition) করতে হবে।

আপনাকে অবশ্যই ডিভাইসটি রিবুট করতে হবে।

পাওয়ার বাটন চেপে ধরে "Restart"-এ ট্যাপ করার মাধ্যমে আপনি GrapheneOS-কে হার্ডওয়্যার পাওয়ার ডাউন (power down) করার নির্দেশ দেন। ঠিক সেই মুহূর্তেই RAM পুরোপুরি মুছে যায় এবং ডিক্রিপশন কি-গুলো উবে যায় (evaporate)। যখন ডিভাইসটি পুনরায় চালু হয় এবং লক স্ক্রিনে এসে আপনার PIN চায়, তখন এটি সম্পূর্ণ অভেদ্য BFU স্টেটে থাকে।

এই কাজটি করতে মাত্র তিন সেকেন্ড সময় লাগে, কিন্তু এটি আপনার ডেটা সিকিউরিটির জন্য গাণিতিক নিশ্চয়তা প্রদান করে। যদি এজেন্ট ডিভাইসটি দাবি করে, তবে আপনি মূলত তাদের হাতে একটি গাণিতিকভাবে সিল করা ইট বা 'brick' তুলে দিচ্ছেন।

The Auto-Reboot Fail-Safe

এলিট অপারেটররা কেবল মানুষের স্মৃতির ওপর নির্ভর করে থাকেন না। আপনাকে হয়তো হঠাৎ করেই আটকে রাখা হতে পারে, অথবা ম্যানুয়াল রিবুট করার জন্য প্রয়োজনীয় তিন সেকেন্ড সময় পাওয়ার আগেই কেউ আপনার হাত থেকে ডিভাইসটি ছোঁ মেরে কেড়ে নিতে পারে।

এই পরিস্থিতি মোকাবিলা করার জন্য GrapheneOS-এ একটি প্রোপ্রাইটারি **Auto-Reboot timer** রয়েছে।

এই ফিচারটি ডিজাইন করা হয়েছে একটি টাইমারের ওপর ভিত্তি করে স্বয়ংক্রিয়ভাবে 'Reboot Rule' প্রয়োগ করার জন্য, যা নিশ্চিত করে যে বাজেয়াপ্ত হওয়া একটি ডিভাইস কোনো এভিডেন্স লকারে (evidence locker) পড়ে থাকার সময় শেষ পর্যন্ত একটি সুরক্ষিত BFU স্টেটে ফিরে যাবে।

Configuration Guide: ১. নেভিগেট করুন **Settings > Security > Auto-reboot**-এ। ২. লক্ষ্য করুন যে GrapheneOS ডিফল্টভাবে এই টাইমারটি ১৮ ঘণ্টার জন্য সেট করে রাখে। ৩. প্রতিকূল কোনো এখতিয়ার বা জুরিসডিকশনের (jurisdictions) ভেতর দিয়ে ভ্রমণের সময় এই টাইমারটি কমিয়ে **৪ ঘণ্টা** বা তারও নিচে নামিয়ে আনুন।

যদি আপনার ডিভাইসটি বাজেয়াপ্ত করা হয় এবং টানা ৪ ঘণ্টা স্ক্রিনটি লক করা অবস্থায় থাকে এবং কোনো PIN এন্টার করা না হয়, তবে GrapheneOS স্বয়ংক্রিয়ভাবে হার্ডওয়্যার শাটডাউন করে দেবে, RAM মুছে ফেলবে এবং রিবুট করে BFU স্টেটে চলে যাবে। এর মাধ্যমে প্রতিপক্ষের ডেটা এক্সট্রাক্ট করার সুযোগ বা 'forensic window of opportunity' সজোরে বন্ধ হয়ে যায়।

ফিজিক্যাল জগতে, আপনার এনক্রিপশন ঠিক ততটাই শক্তিশালী যতটা আপনার RAM-এর অবস্থা শক্তিশালী। BFU স্টেটটি আয়ত্ত করুন। Reboot Rule প্রয়োগ করুন। কি (keys)-গুলোর নিয়ন্ত্রণ নিজের হাতে রাখুন।

২.১৬ The Deadman's Switch: Auto-Reboot এবং Key Wiping

ফিজিক্যাল সিকিউরিটি আর্কিটেকচারে, "Deadman's Switch" হলো একটি ফেইল-সেইফ (fail-safe) মেকানিজম বা স্বয়ংক্রিয় নিরাপত্তাব্যবস্থা, যা এমনভাবে ডিজাইন করা হয় যাতে হিউম্যান অপারেটর যদি অক্ষম (incapacitated) বা নিষ্ক্রিয় হয়ে যান, অথবা তাকে জোরপূর্বক নিয়ন্ত্রণ থেকে সরিয়ে দেওয়া হয়, তখন এটি স্বয়ংক্রিয়ভাবে ট্রিগার (trigger) বা চালু হয়। এটি 'নেগেটিভ অ্যাকশন' বা নেতিবাচক ক্রিয়ার নীতির ওপর কাজ করে: সিস্টেমটি তখন এর ডিফেন্সিভ প্রোটোকল বা প্রতিরক্ষামূলক ব্যবস্থা এক্সিকিউট করে না যখন আপনি তাকে নির্দেশ দেন; বরং ঠিক তখনই করে, যখন আপনি তাকে নির্দেশ দিতে *ব্যর্থ* হন।

মোবাইল ফরেনসিকের (mobile forensics) জগতে, সময় হলো প্রতিপক্ষের সবচেয়ে বড় হাতিয়ার বা সম্পদ।

যখন কোনো প্রতিকূল বা শত্রু পক্ষ—তা সে কোনো গোয়েন্দা সংস্থা, বর্ডার প্যাট্রোল, অথবা কর্পোরেট গুপ্তচর দলই হোক না কেন—আপনার ডিভাইসটি বাজেয়াপ্ত করে, তখন তারা খুব কম সময়ই রাস্তায় দাঁড়িয়ে তাৎক্ষণিকভাবে ডেটা এক্সট্রাক্ট করে। রিমোট ওয়াইপ (remote wipe) কমান্ড ব্লক করার জন্য ডিভাইসটিকে একটি ফ্যারাডে ব্যাগে (Faraday bag) রাখা হয়, একটি এভিডেন্স চেইনে (evidence chain) লগ বা নথিভুক্ত করা হয়, একটি সুরক্ষিত ফরেনসিক ল্যাবরেটরিতে পরিবহন করে নিয়ে যাওয়া হয় এবং Cellebrite-এর মতো কোনো এক্সট্রাকশন মেশিনের সাথে যুক্ত করার অপেক্ষায় একটি সারিতে বা কিউতে (queue) রেখে দেওয়া হয়।

এই সম্পূর্ণ প্রক্রিয়ায় কয়েক ঘণ্টা থেকে শুরু করে কয়েক দিন পর্যন্ত সময় লাগতে পারে। পূর্ববর্তী সেকশনে যেমনটি আমরা জেনেছি, যদি আপনার ডিভাইসটিকে চালু থাকা অবস্থায় এবং পূর্বে অন্তত একবার আনলক করা অবস্থায় বাজেয়াপ্ত করা হয়ে থাকে, তবে এটি চরম ঝুঁকিপূর্ণ AFU (After First Unlock) স্টেটে অবস্থান করে। এর ডিক্রিপশন কি-গুলো (decryption keys) তখন Random Access Memory (RAM)-এর ভেতরে 'হট' (hot) বা সক্রিয় অবস্থায় থাকে এবং ফরেনসিক টেকনিশিয়ানের সেগুলো এক্সট্রাক্ট করার অপেক্ষায় থাকে।

ফরেনসিকের এই সময়সীমাকে (forensic timeline) পুরোপুরি নিষ্ক্রিয় করার জন্য, GrapheneOS একটি ক্রিপ্টোগ্রাফিক Deadman's Switch ডিপ্লয় (deploy) বা মোতায়েন করে: আর সেটি হলো **Auto-Reboot Timer**।

The Mechanics of Auto-Reboot

বিশ্বের একমাত্র মোবাইল অপারেটিং সিস্টেম হিসেবে GrapheneOS সরাসরি তার মূল সিকিউরিটি ফ্রেমওয়ার্কে (core security framework) একটি স্বয়ংক্রিয় এবং সময়-ভিত্তিক (time-based) রিবুট সিকোয়েন্স ইন্টিগ্রেট (integrate) করে।

এর পেছনের যুক্তিটি চমৎকার এবং অত্যন্ত সাধারণ। অপারেটিং সিস্টেমটি ডিভাইসের সাথে আপনার ইন্টার‍্যাকশন বা যোগাযোগটি প্রতিনিয়ত মনিটর করে। আরও নির্দিষ্টভাবে বললে, এটি আপনার সফল অথেনটিকেশন (সঠিক PIN বা পাসওয়ার্ড এন্টার করা) মনিটর করে। যদি ডিভাইসটি অলসভাবে বা আইডল (idle) অবস্থায় পড়ে থাকে এবং ইউজারের সেট করে দেওয়া কাউন্টডাউন টাইমারের মধ্যে সঠিক PIN এন্টার করা না হয়, তবে অপারেটিং সিস্টেম সম্ভাব্য সবচেয়ে খারাপ পরিস্থিতিটি ধরে নেয়: এটি ধরে নেয় যে অপারেটরটি কম্প্রোমাইজড বা হ্যাক হয়েছেন, অক্ষম হয়ে পড়েছেন, অথবা হার্ডওয়্যার থেকে বিচ্ছিন্ন হয়ে গেছেন।

কাউন্টডাউন টাইমারটি শূন্য (zero)-তে পৌঁছানোর ঠিক মুহূর্তেই, GrapheneOS একটি হার্ড (hard) এবং স্বয়ংক্রিয় শাটডাউন ও রিবুট সিকোয়েন্স এক্সিকিউট করে।

The Cryptographic Wipe

Auto-Reboot ফিচারের মূল চাতুর্য বা বুদ্ধিমত্তা শুধু রিবুট করার মাঝেই সীমাবদ্ধ নয়, বরং এই প্রক্রিয়ার সময় যে ক্রিপ্টোগ্রাফিক স্যানিটাইজেশন (cryptographic sanitization) বা পরিচ্ছন্নতা ঘটে, সেখানেই এর প্রকৃত রূপটি ফুটে ওঠে।

টাইমারটি যখন ট্রিগার (trigger) হয়, তখন GrapheneOS শুধুমাত্র গ্রাফিকাল ইন্টারফেসটি রিস্টার্ট করে না। এটি সিস্টেম RAM-কে সক্রিয়ভাবে মুছে ফেলার (wipe) জন্য একটি লো-লেভেল কমান্ড (low-level command) এক্সিকিউট করে। যেসব ডিক্রিপশন কি AFU স্টেটে আপনার ফাইল সিস্টেমকে অ্যাক্সেসযোগ্য করে রেখেছিল, সেগুলোকে জোরপূর্বক জিরো-আউট (zeroed out) করা হয় এবং হার্ডওয়্যার রেজিস্টার (hardware registers) থেকে উধাও বা ইভাপোরেট (evaporate) করে দেওয়া হয়।

ডিভাইসটি যখন পুনরায় চালু হয় এবং প্রাথমিক লক স্ক্রিনে এসে থামে, তখন এটি সফলভাবে সম্পূর্ণ অভেদ্য BFU (Before First Unlock) স্টেটে ট্রানজিশন করে ফিরে যায়। আপনার ডেটা তখন গাণিতিকভাবে সিল (sealed) বা তালাবদ্ধ অবস্থায় বিশ্রাম নেয় (at rest)।

ডিভাইসটি বাজেয়াপ্ত করার দশ ঘণ্টা পর যদি কোনো ফরেনসিক টেকনিশিয়ান এভিডেন্স লকার থেকে আপনার ডিভাইসটি বের করে আনেন, তবে বড্ড দেরি হয়ে যায়। Deadman's Switch ততক্ষণে ট্রিগার হয়ে গেছে। কি (keys)-গুলো হারিয়ে গেছে এবং এক্সট্রাকশন মেশিনটি মূলত গাণিতিকভাবে সিল করা একটি ইটের (brick) মুখোমুখি এসে দাঁড়ায়।

Defeating the "Cold Boot" Attack

Auto-Reboot টাইমারের প্রয়োজনীয়তা সঠিকভাবে উপলব্ধি করতে হলে, আমাদের অবশ্যই ফরেনসিক অস্ত্রাগারের সবচেয়ে অত্যাধুনিক এবং হাই-টিয়ার (high-tier) ফিজিক্যাল অ্যাটাকগুলোর একটি নিয়ে আলোচনা করতে হবে: যাকে বলা হয় **Cold Boot Attack**।

RAM হলো এক ধরণের ভোলাটাইল বা অস্থায়ী মেমোরি (volatile memory)। কোনো কম্পিউটারের পাওয়ার বন্ধ হয়ে গেলে, RAM-এ সংরক্ষিত ডেটা (আপনার এনক্রিপশন কি-গুলো সহ) তাৎক্ষণিকভাবে হারিয়ে যাওয়ার কথা। কিন্তু, পদার্থবিদ এবং সাইবার সিকিউরিটি গবেষকরা RAM চিপের একটি মারাত্মক ফিজিক্যাল ত্রুটি আবিষ্কার করেছেন: আর সেটি হলো, মেমোরি ডিকে (memory decay) বা স্মৃতি হারিয়ে যাওয়া সরাসরি তাপমাত্রার সাথে সম্পর্কিত।

যদি কোনো ডিভাইস AFU স্টেটে থাকে, তবে কি (keys)-গুলো RAM-এই অবস্থান করে। যদি কোনো প্রতিপক্ষ হঠাৎ করে ডিভাইসের ব্যাটারি আনপ্লাগ করে দেয়, তবে ভগ্নাংশ সেকেন্ডের মধ্যেই কি-গুলো মুছে যায়। কিন্তু, যদি প্রতিপক্ষ পাওয়ার কাট করার *আগে* RAM চিপের ওপর কম্প্রেসড এয়ার (compressed air) বা লিকুইড নাইট্রোজেন (liquid nitrogen) স্প্রে করে তাপমাত্রা সাব-জিরো (sub-zero) বা শূন্যের নিচে নামিয়ে আনে, তবে ফিজিক্যাল ইলেকট্রনগুলো সেই জায়গাতেই জমে বা ফ্রিজ (freeze) হয়ে যায়। এর ফলে মেমোরি ডিকে-র প্রক্রিয়াটি মারাত্মকভাবে ধীর হয়ে যায় এবং মিলিসেকেন্ডের বদলে কয়েক মিনিট পর্যন্ত সময় নেয়।

একজন অত্যাধুনিক প্রতিপক্ষ RAM-কে ফ্রিজ করতে পারে, আপনার ডিভাইসের মাদারবোর্ড থেকে চিপটিকে ফিজিক্যালি তুলে নিতে পারে, সেটিকে একটি বিশেষ রিডিং মেশিনে বসাতে পারে এবং বরফ গলার আগেই সেই জমে থাকা এনক্রিপশন কি-গুলো এক্সট্রাক্ট করে নিতে পারে।

Auto-Reboot টাইমার মূলত এই Cold Boot অ্যাটাকের সরাসরি একটি কাউন্টারমেজার (countermeasure) বা প্রতিরোধ ব্যবস্থা।

ডিভাইসটি AFU স্টেটে যতটা সময় কাটায়, তাকে অ্যাগ্রেসিভভাবে বা অত্যন্ত কঠোরভাবে কমিয়ে আনার মাধ্যমে, আপনি আক্রমণকারীর ফ্রিজ এক্সিকিউট করার সুযোগের জানালাটিকে বা 'window of opportunity' মারাত্মকভাবে সংকুচিত করে দেন। প্রতিপক্ষের ট্রান্সপোর্ট ভেহিকল বা পরিবহন যানে বসে থাকার সময়ই যদি আপনার ডিভাইসটি রিবুট হয়ে যায় এবং RAM থেকে কি-গুলোকে পার্জ (purges) বা পুরোপুরি মুছে ফেলে, তবে ল্যাবরেটরিতে নেওয়ার পর কোনো লিকুইড নাইট্রোজেন অ্যাটাক করেও তারা আর কিছুই পাবে না। যেগুলো আগে থেকেই গাণিতিকভাবে জিরো-আউট (zeroed out) হয়ে গেছে, সেই কি-গুলোকে আপনি আর ফ্রিজ করতে পারবেন না।

Configuring the Deadman's Switch

ডিফল্টভাবে, GrapheneOS বেশ রক্ষণশীলভাবে Auto-Reboot টাইমারটিকে ১৮ ঘণ্টার জন্য সেট করে রাখে। এটি হলো সাধারণ ব্যবহারকারীদের জন্য ডিজাইন করা একটি বেইজলাইন বা প্রাথমিক নিরাপত্তা—যা নিশ্চিত করে যে কোনো ফোন হারিয়ে গেলে বা চুরি হয়ে গেলে, তা প্রতিদিনের কোনো বিরক্তি সৃষ্টি না করেই পরদিন সকালের মধ্যে নিজেকে সুরক্ষিত করে ফেলবে।

তবে, এলিট অপারেটরদের অবশ্যই তাদের নিজস্ব থ্রেট মডেল (threat model) অনুযায়ী এই টাইমারটি কনফিগার করা উচিত।

১. নেভিগেট করুন **Settings > Security > Auto-reboot**-এ। ২. অপশনগুলোর রেঞ্জ থেকে (১৮ ঘণ্টা থেকে শুরু করে ১০ মিনিট পর্যন্ত) যেকোনো একটি টাইমার ভ্যালু সিলেক্ট করুন।

The Threat Matrix Calibration:

- **18 Hours (Default):** লো-থ্রেট (low-threat) বা কম ঝুঁকিপূর্ণ এবং ঘরোয়া পরিবেশের জন্য উপযোগী, যেখানে মূল চিন্তার বিষয় হলো কোনো সাধারণ নাগরিকের হাতে হারিয়ে যাওয়া ডিভাইসটি পড়া।
- **4 Hours:** এটি মূলত কর্পোরেট সিকিউরিটি প্রফেশনাল এবং সাংবাদিকদের জন্য রেকমেন্ড করা বা প্রস্তাবিত বেইজলাইন। এটি আইন প্রয়োগকারী সংস্থার বাজেয়াপ্তকরণের বিরুদ্ধে একটি সংক্ষিপ্ত সময়সীমা বা টাইট উইন্ডো (tight window) প্রদান করে, আবার একই সাথে এটি এতটাই সুবিধাজনক যে টাইমারটি ট্রিগার হওয়ার আগেই আপনি হয়তো স্বাভাবিকভাবেই দিনের বেলা ফোনটি আনলক করে ফেলবেন।
- **1 Hour:** যেসব অপারেটর কোনো প্রতিকূল পরিবেশের (যেমন রাজনৈতিক বিক্ষোভ) ভেতর দিয়ে চলাফেরা করছেন, যেখানে ছোঁ মেরে কেড়ে নেওয়া বা তাৎক্ষণিক পুলিশি আটকের (immediate police detentions) চরম সম্ভাবনা রয়েছে, তাদের জন্য। আপনি যদি আটকা পড়েন, তবে ডিভাইসটি পুলিশ স্টেশনে বা প্রিসিঙ্কটে (precinct) পৌঁছানোর আগেই নিজেকে সুরক্ষিত করে ফেলবে।
- **10 Minutes:** এটি হলো অ্যাবসলিউট ম্যাক্সিমাম (absolute maximum) বা সর্বোচ্চ পর্যায়ের নিরাপত্তা কৌশল। এটি শুধুমাত্র সেসব অপারেটরদের জন্য সংরক্ষিত যারা কোনো অ্যাক্টিভ কনফ্লিক্ট জোন (active conflict zones) বা যুদ্ধক্ষেত্রে অবস্থান করছেন অথবা চরম শত্রুতাপূর্ণ বর্ডার অতিক্রম করছেন। ডিভাইসটি আপনার ফিজিক্যাল পজেশন বা নিয়ন্ত্রণ থেকে হাতছাড়া হওয়ার প্রায় সাথে সাথেই এটি নিজেকে সিল বা তালাবদ্ধ করে ফেলে।

নিরাপত্তা হলো নিয়ন্ত্রণেরই একটি পরিমাপ। Auto-Reboot টাইমার এটি নিশ্চিত করে যে, এমনকি যখন আপনি আপনার হার্ডওয়্যারের ওপর ফিজিক্যাল নিয়ন্ত্রণ হারিয়ে ফেলেন, তখনও আপনি আপনার ক্রিপ্টোগ্রাফির ওপর গাণিতিক নিয়ন্ত্রণ হারাবেন না। Deadman's Switch নিশ্চিত করে যে দুর্গটি নিজেই নিজেকে রক্ষা করতে সক্ষম।

২.১৭ The Nuclear Option: Duress PINs

আমরা এই পুরো বইটি মূলত ম্যাথমেটিকাল ডিফেন্স বা গাণিতিক প্রতিরক্ষাব্যবস্থা নির্মাণের জন্যই উৎসর্গ করেছি। আমরা ক্রিপ্টোগ্রাফিক স্যান্ডবক্স (cryptographic sandboxes), এনক্রিপ্টেড নেটওয়ার্ক রাউটিং (encrypted network routing) এবং হার্ডওয়্যার আইসোলেশন (hardware isolation) বাস্তবায়ন করেছি। এই ডিফেন্সগুলো এমনভাবে ডিজাইন করা হয়েছে যেন সেগুলো ডিজিটাল প্রতিপক্ষ, ফরেনসিক মেশিন এবং জিরো-ডে এক্সপ্লয়েটের (zero-day exploits) কাছে পুরোপুরি অভেদ্য (impenetrable) থাকে।

তবে, ক্রিপ্টোগ্রাফির চূড়ান্ত সীমাবদ্ধতার মুখোমুখি আমাদের হতেই হবে: আর তা হলো, গণিত আপনাকে শারীরিক সহিংসতার হাত থেকে রক্ষা করতে পারে না।

ইন্টেলিজেন্স বা গোয়েন্দা এবং সাইবার সিকিউরিটি কমিউনিটিতে এই দুর্বলতাটিকে ভয়ংকরভাবে "**rubber-hose cryptanalysis**" বা 'রাবার-হোস ক্রিপ্টানালাইসিস' বলে উল্লেখ করা হয়। আপনার ডিভাইসটি 256-bit AES এনক্রিপশন এবং Titan M2 হার্ডওয়্যার সিকিউরিটি মডিউল (hardware security module) ব্যবহার করে কি না, তাতে কিছুই যায় আসে না, যদি প্রতিপক্ষ স্রেফ আপনাকে একটি চেয়ারের সাথে বেঁধে রাবারের পাইপ দিয়ে পিটিয়ে আপনার মুখ থেকে পাসওয়ার্ড বের করে নেয়।

প্রতিকূল রাষ্ট্র-সমর্থিত হ্যাকার (state actors), ড্রাগ কার্টেল (cartels), বা দুর্নীতিগ্রস্ত বর্ডার গার্ডদের মুখোমুখি হলে, আপনি এমন একটি পরিস্থিতিতে পড়তে পারেন যেখানে আপনার পাসওয়ার্ড সারেন্ডার করতে বা দিতে অস্বীকৃতি জানালে আপনাকে শারীরিক ক্ষতি, নির্যাতন, বা অনির্দিষ্টকালের আটকের শিকার হতে হবে। তখন আপনার থ্রেট মডেলটি (threat model) আপনার ডেটা রক্ষার বদলে আপনার জীবন রক্ষার দিকে স্থানান্তরিত হয়ে যায়।

এই চরম দৃশ্যকল্পগুলোতে, GrapheneOS চূড়ান্ত এবং অপরিবর্তনীয় একটি কাউন্টারমেজার (countermeasure) বা প্রতিরোধ ব্যবস্থা প্রদান করে: আর সেটি হলো **Duress PIN**।

The Mechanics of the Nuclear Option

Duress PIN (বা Duress Password) হলো দ্বিতীয় একটি অত্যন্ত সুনির্দিষ্ট পাসওয়ার্ড, যা আপনি মূলত প্রতারণা (deception) এবং ধ্বংসের (destruction) উদ্দেশ্যেই কনফিগার করে থাকেন।

কল্পনা করুন, আপনাকে কোনো প্রতিকূল বা বৈরী এখতিয়ারে (hostile jurisdiction) আটকে রাখা হয়েছে। প্রতিপক্ষ আপনার সামনে আপনার ডিভাইসটি রাখে এবং PIN দাবি করে। আপনি তাদের দাবি মেনে নিলেন। আপনি অত্যন্ত শান্তভাবে লক স্ক্রিনে একটি PIN টাইপ করলেন।

আপনাকে পর্যবেক্ষণ করা প্রতিপক্ষের কাছে মনে হবে যে আপনি সারেন্ডার করেছেন। আপনি একটি কোড টাইপ করেছেন এবং ডিভাইসটি সেটি গ্রহণ করেছে। কিন্তু, আপনি যে কোডটি টাইপ করেছেন সেটি আপনার আসল আনলকিং PIN ছিল না; সেটি ছিল আপনার Duress PIN।

Duress PIN এন্টার করার মুহূর্তেই, GrapheneOS ডিভাইসটিকে আনলক করে না। এটি কোনো এরর মেসেজ (error message) দেখায় না, এবং কোনো লোডিং স্ক্রিনও প্রদর্শন করে না। এর বদলে, অপারেটিং সিস্টেমটি নীরবে এবং তাৎক্ষণিকভাবে সরাসরি **Titan M2** হার্ডওয়্যার সিকিউরিটি মডিউলের কাছে একটি টার্মিনেশন কমান্ড (termination command) পাঠিয়ে দেয়।

Titan M2 চিপ তখন একটি **Cryptographic Wipe** এক্সিকিউট করে।

এটি কষ্ট করে আপনার ছবিগুলো ডিলিট করার বা শূন্য (zeroes) দিয়ে আপনার ফাইলগুলোকে ওভাররাইট করার চেষ্টা করে না—কারণ এই প্রক্রিয়ায় কয়েক মিনিট সময় লাগতে পারে এবং প্রতিপক্ষ চাইলে ব্যাটারি খুলে ফেলে তা বিঘ্নিতও করতে পারে। এর বদলে, Titan M2 তাৎক্ষণিকভাবে এবং চিরতরে সেই মাস্টার হার্ডওয়্যার কি (Weaver token) মুছে ফেলে যা আপনার ফাইল সিস্টেমকে ডিক্রিপ্ট করে।

মিলিসেকেন্ডের ভগ্নাংশের মধ্যেই, আপনার ডেটার সাথে যুক্ত গাণিতিক সেতুটি (mathematical bridge) পুড়ে ছাই হয়ে যায়। আপনার ফাইল, আপনার কন্টাক্ট, আপনার এনক্রিপ্টেড মেসেজ এবং আপনার ইউজার প্রোফাইলগুলো তাৎক্ষণিকভাবে একটি অপঠনযোগ্য, চিরতরে অ্যাক্সেস অযোগ্য এবং র‍্যান্ডমাইজড নয়েজের (randomized noise) সাগরে পরিণত হয়।

কি (key) ধ্বংস হওয়ার ঠিক পরপরই, ডিভাইসটি জোরপূর্বক রিবুট হয়। এটি যখন আবার চালু হয়, তখন প্রতিপক্ষ স্রেফ প্রাথমিক, ফ্যাক্টরি-ফ্রেশ বা একেবারে নতুন একটি Android সেটআপ স্ক্রিনের দিকে তাকিয়ে থাকে। আপনার ডেটা ততক্ষণে পুরোপুরি নিশ্চিহ্ন। ডিভাইসটি একটি ব্ল্যাঙ্ক স্লেট (blank slate) বা খালি পৃষ্ঠায় পরিণত হয়ে গেছে। আপনি হার্ডওয়্যার সারেন্ডার করেছেন ঠিকই, কিন্তু আপনি সফলভাবে আপনার ইন্টেলিজেন্স রক্ষা করেছেন।

Configuring the Duress PIN

যেহেতু Duress PIN হলো শেষ আশ্রয়ের একটি অস্ত্র বা 'weapon of last resort', তাই এর কনফিগারেশন অত্যন্ত সতর্কতার সাথে পরিচালনা করতে হবে।

১. নেভিগেট করুন **Settings** > **Security**-এ। ২. **Duress PIN/Password** অপশনটিতে ট্যাপ করুন। ৩. যখন আপনার পরিচয় ভেরিফাই করতে বলা হবে, তখন আপনার *আসল* (actual) ডিভাইস PIN-টি এন্টার করুন। ৪. এরপর আপনার নতুন Duress PIN-টি এন্টার করুন।

> ***OpSec Rule for Duress PINs:*** *আপনার Duress PIN-টি অবশ্যই আপনার আসল PIN থেকে সম্পূর্ণ আলাদা হতে হবে, তবে চরম মানসিক চাপের মধ্যেও যেন এটি সহজেই মনে রাখা যায় এমন হতে হবে। আপনার আসল PIN যদি `849201` হয়, তবে আপনার Duress PIN কখনোই `849202` বানাবেন না—কারণ প্রতিপক্ষ আপনার পাসওয়ার্ড অনুমান করার সময় ভুলবশত এই ওয়াইপ (wipe) কমান্ডটি ট্রিগার করে ফেলতে পারে। সম্পূর্ণ আলাদা এবং অত্যন্ত সহজেই মনে রাখা যায় এমন একটি সিকোয়েন্স (sequence) বেছে নিন।*

The Warning: Irreversible Destruction

একটি Duress PIN বাস্তবায়ন করার জন্য মারাত্মক মানসিক পরিবর্তনের প্রয়োজন। মূলত আপনি আপনার পকেটে একটি সেলফ-ডেস্ট্রাক্ট মেকানিজম (self-destruct mechanism) বা আত্মবিধ্বংসী ব্যবস্থা ইনস্টল করছেন।

আপনাকে অবশ্যই এই ফিচারের পরম চূড়ান্ত অবস্থাটি বুঝতে হবে।

- **There is no recovery.** একটি ক্রিপ্টোগ্রাফিক ওয়াইপ (cryptographic wipe) বা মুছে ফেলা ডেটা কখনোই রিকভার বা পুনরুদ্ধার করা সম্ভব নয়। আপনি নিজে, GrapheneOS ডেভেলপার, এমনকি পৃথিবীর সবচেয়ে উন্নত ডিজিটাল ফরেনসিক ল্যাবরেটরিও এটি করতে পারবে না। একবার Titan M2 চিপের মাধ্যমে Weaver টোকেন (Weaver token) ধ্বংস হয়ে গেলে, এর গাণিতিক ফলাফল একেবারেই চূড়ান্ত বা অপরিবর্তনীয় হয়ে যায়।
- **Do not test it.** অনেক নতুন অপারেটর একটি Duress PIN কনফিগার করে এবং তারপর "এটি কীভাবে কাজ করে তা দেখার জন্য স্রেফ পরীক্ষা করে দেখি" এই ভেবে এটি টেস্ট করার সিদ্ধান্ত নেন, তারা ধরে নেন যে তারা সহজেই আবার লগ ইন করতে পারবেন বা একটি দ্রুত ব্যাকআপ রিস্টোর (restore) করে নিতে পারবেন। আপনি যদি আপনার

Duress PIN টেস্ট বা পরীক্ষা করেন, তবে আপনার ডিভাইসটি তাৎক্ষণিকভাবে ফ্যাক্টরি রিসেট (factory reset) হয়ে যাবে। ডিভাইসের বাইরে ব্যাকআপ করে না রাখা সমস্ত ডেটা চিরতরে ধ্বংস বা নিশ্চিহ্ন হয়ে যাবে।

- **Beware of accidental triggering.** আপনার যদি এমন সন্তান থাকে যারা প্রায়শই আপনার ফোন নিয়ে খেলা করে, অথবা আপনি যদি ঘুমের ঘোরে টাইপ করার বা ভারী নেশায় মদ্যপ থাকার প্রতি আসক্ত হন, তবে আপনাকে অবশ্যই গুরুত্ব সহকারে মূল্যায়ন করতে হবে যে আপনার থ্রেট মডেলের (threat model) জন্য একটি Duress PIN আদৌ উপযুক্ত কি না। ভুলবশত একবার এটি এন্টার করলেই আপনার সম্পূর্ণ ডেটা চিরতরে হারিয়ে যাবে।

Duress PIN হলো নিউক্লিয়ার অপশন বা শেষ অস্ত্র। এটি একটি ট্যাকটিকাল স্বীকৃতি যে হার্ডওয়্যার পরিবর্তনযোগ্য, কিন্তু ইন্টেলিজেন্স বা ডেটা নয়। এই মেকানিজমটি কনফিগার করার মাধ্যমে, আপনি এটি নিশ্চিত করেন যে এমনকি অপ্রতিরোধ্য ফিজিক্যাল জোর-জবরদস্তির মুখেও, আপনার ডেটার চূড়ান্ত নিয়তির ওপর আপনি পরম বা নিরঙ্কুশ নিয়ন্ত্রণ ধরে রাখবেন।

২.১৮ Locking the Gates: USB-C Port Control

কোনো মোবাইল ডিভাইসের অ্যাটাক সারফেস (attack surface) বিশ্লেষণ করার সময়, আমরা প্রায়শই অদৃশ্য ভেক্টরগুলোর (invisible vectors) ওপর তীব্রভাবে ফোকাস করি: যেমন সেলুলার রেডিও ওয়েভ, Wi-Fi কানেকশন এবং ইন্টারনেট রাউটিং প্রোটোকল। কিন্তু আপনার ডিভাইসের সবচেয়ে বড় এবং সবচেয়ে ঝুঁকিপূর্ণ প্রবেশদ্বারটি মোটেও অদৃশ্য কিছু নয়। এটি হলো আপনার ফোনের ঠিক নিচে থাকা ফিজিক্যাল, আয়তাকার গর্তটি।

USB-C পোর্টটি মূলত একটি হাই-স্পিড ডেটা পাইপলাইন (high-speed data pipeline) হিসেবে কাজ করার জন্য ডিজাইন করা হয়েছে। একটি স্ট্যান্ডার্ড বা গতানুগতিক কমার্শিয়াল অপারেটিং সিস্টেমে, এই পাইপলাইনটি অত্যন্ত প্রমিসকিউয়াস (promiscuous) বা বাছ-বিচারহীন আচরণ করে। এটি যেকোনো কিছুর সাথে কানেক্ট করতে, ডেটা শেয়ার করতে এবং ইনকামিং কমান্ড (incoming commands) গ্রহণ করার জন্য সর্বদা উন্মুখ হয়ে থাকে। ফিজিক্যালি আপনার ডিভাইসের নাগাল পাওয়া যেকোনো প্রতিপক্ষের কাছে, এই পোর্টটিই হলো এক্সপ্লয়েটেশন বা শোষণের প্রধান ভেক্টর।

হার্ডওয়্যারকে সত্যিকার অর্থে সুরক্ষিত করতে হলে, আমাদের অবশ্যই এই ফিজিক্যাল গেটগুলো বা প্রবেশদ্বারগুলোতে তালা লাগাতে হবে।

The Physical Attack Vectors

যে প্রতিপক্ষ আপনার ডিভাইসে ফিজিক্যালি অ্যাক্সেস পায়—অথবা ধোঁকা দিয়ে তাদের হার্ডওয়্যারে আপনার ডিভাইসটি প্লাগ (plug) করতে বাধ্য করে—তারা ধ্বংসাত্মক অ্যাটাক বা আক্রমণ চালানোর জন্য USB-C পোর্টটিকে এক্সপ্লয়েট করতে পারে।

১. Malicious Charging Stations (Juice Jacking): ধরে নিন, আপনি একটি আন্তর্জাতিক বিমানবন্দরের ভেতর দিয়ে হেঁটে যাচ্ছেন এবং আপনার ফোনের ব্যাটারি প্রায় শেষ হয়ে আসছে। আপনি একটি পাবলিক চার্জিং কিয়স্কে (kiosk) আপনার ফোনটি প্লাগ করলেন। আপনার অজান্তেই, কিয়স্কটি মূলত কম্প্রোমাইজড বা হ্যাক করা একটি স্টেশন। একটি স্ট্যান্ডার্ড USB ক্যাবলে পাওয়ার পিন (power pins) এবং ডেটা পিন (data pins) উভয়ই থাকে। যখন আপনি এটি প্লাগ করেন, তখন এই ক্ষতিকর কিয়স্কটি ডেটা পিনগুলো ব্যবহার করে আপনার ডিভাইসে নীরবে ম্যালওয়্যার ইনস্টল করার অথবা আপনার ফটো গ্যালারিটি চুরি বা সাইফন (syphon) করার চেষ্টা করে, আর এর পাশাপাশি পাওয়ার পিনগুলো আপনার ব্যাটারি চার্জ করতে থাকে।

২. Keystroke Injection (USB Rubber Duckies): অত্যন্ত বুদ্ধিমান কোনো প্রতিপক্ষ হয়তো আপনার ব্যক্তিগত চার্জিং ক্যাবলটি পরিবর্তন করে এমন একটি ডিভাইস রেখে দিতে পারে যা "USB Rubber Ducky" অথবা "O.MG Cable" নামে পরিচিত। মানুষের চোখে, এটি দেখতে একেবারে সাধারণ একটি USB-C চার্জিং কর্ডের (cord) মতোই মনে

হয়। কিন্তু, প্লাস্টিকের কানেক্টরের গভীরে লুকানো থাকে একটি মাইক্রো-কম্পিউটার (micro-computer)। যখন আপনি এটিকে আপনার ফোনে প্লাগ করেন, তখন আপনার ফোন এটিকে কোনো ক্যাবল হিসেবে দেখে না; এটি ডিভাইসটিকে একটি ফিজিক্যাল কিবোর্ড (physical keyboard) হিসেবে রেজিস্টার করে। ক্যাবলটি তখন প্রতি মিনিটে হাজার হাজার শব্দের গতিতে দ্রুতগতিতে আগে থেকে প্রোগ্রাম করা কিস্ট্রোকের (keystrokes) একটি সিকোয়েন্স টাইপ করতে শুরু করে, যা আপনি কোনো প্রতিক্রিয়া দেখানোর আগেই মেনু বাইপাস করে, একটি টার্মিনাল খোলে এবং একটি ম্যালিসিয়াস পেলোড (malicious payload) এক্সিকিউট করে দেয়।

৩. **Forensic Extraction Boxes:** আমাদের ট্রাভেল সিকিউরিটি সেকশনগুলোতে আলোচনা করা হয়েছে যে, বর্ডার এজেন্ট বা গোয়েন্দা কর্মীরা যদি আপনার ডিভাইস বাজেয়াপ্ত করে, তবে তাদের প্রথম কাজই হলো আপনার USB-C পোর্টে একটি অত্যন্ত বিশেষায়িত ফরেনসিক ক্যাবল (যেমন GrayKey বা Cellebrite টার্মিনাল) প্লাগ করা। এই মেশিনগুলো লক স্ক্রিন বাইপাস করার এবং RAM-এ অ্যাক্সেস নেওয়ার জন্য USB ড্রাইভারগুলোতে কোনো দুর্বলতা খুঁজে পাওয়ার আশায় পোর্টটির ওপর শত শত জিরো-ডে এক্সপ্লয়েট (zero-day exploits) দিয়ে অ্যাগ্রেসিভভাবে বা অত্যন্ত কঠোরভাবে আক্রমণ চালায়।

The GrapheneOS Defense: Dynamic Port Control

এই অ্যাটাকগুলো থেকে আপনাকে রক্ষা করার জন্য স্ট্যান্ডার্ড Android মূলত সফটওয়্যার পপ-আপের (software pop-ups) ওপর নির্ভর করে। আপনি যখন কোনো ম্যালিসিয়াস বা ক্ষতিকর কিয়স্কে প্লাগ করেন, তখন স্ট্যান্ডার্ড Android হয়তো একটি প্রম্পট (prompt) দেখাবে, যেখানে লেখা থাকবে "Allow access to device data?"। আপনি যদি অর্ধ-ঘুমন্ত অবস্থায় ভুলবশত "Yes"-এ ট্যাপ করে বসেন, তবে আপনি কম্প্রোমাইজড হয়ে যাবেন।

GrapheneOS মানুষের এই নিখুঁত হওয়ার প্রত্যাশার ওপর নির্ভরতাকে সম্পূর্ণরূপে প্রত্যাখ্যান করে। এর বদলে এটি **USB-C Port Control** নামে একটি অত্যন্ত গভীরভাবে ইন্টিগ্রেটেড এবং হার্ডওয়্যার-লেভেলের (hardware-level) ডিফেন্স ডিপ্লয় বা মোতায়েন করে।

ডিফল্টভাবে, GrapheneOS এমনভাবে USB-C পোর্টটি কনফিগার করে রাখে যেন এটি **Charging-only when locked** নামক একটি মোডে (mode) কাজ করে।

এর পেছনের যুক্তিটি চমৎকার এবং আপোসহীন (uncompromising)। ঠিক যেই মিলিসেকেন্ডে আপনার স্ক্রিনটি অন্ধকার হয়ে যায় এবং ডিভাইসটি লক হয়ে যায়, ঠিক সেই মুহূর্তে GrapheneOS কার্নেল (kernel) গাণিতিকভাবে USB-C পোর্টের ডেটা লাইনগুলো (data lines) বিচ্ছিন্ন বা সিভার (sever) করে দেয়। অপারেটিং সিস্টেমটি কোনো USB ড্রাইভ মাউন্ট

(mount) করতে অস্বীকার করে, ইনকামিং কিবোর্ড চিনতে অস্বীকার করে এবং ফরেনসিক এক্সট্রাকশন মেশিনগুলোকে পুরোপুরি ইগনোর (ignore) করে। পোর্টটি মূলত একটি ডাম্ব পাওয়ার রিসেপ্টাকলে (dumb power receptacle) পরিণত হয়। এটি ব্যাটারি চার্জ করার জন্য বিদ্যুৎ গ্রহণ করবে ঠিকই, কিন্তু এর ডেটা পাইপলাইন তখন কনক্রিট দিয়ে সিল করা থাকবে।

বর্ডার এজেন্টরা যদি স্ক্রিন লক থাকা অবস্থায় আপনার ডিভাইসটি বাজেয়াপ্ত করে এবং এটিকে কোনো Cellebrite মেশিনে প্লাগ করে, তবে মেশিনটি এর ইনিশিয়াল হ্যান্ডশেকও (initial handshake) এক্সিকিউট করতে পারে না। আপনার USB পোর্টটি তখন সেই মেশিনের অস্তিত্বকেই অস্বীকার করে।

Extreme Lockdown Configuration

উচ্চ-ঝুঁকিপূর্ণ বা হাই-থ্রেট পরিবেশে কাজ করা এলিট অপারেটরদের জন্য, GrapheneOS এই ডিফেন্স বা প্রতিরক্ষাব্যবস্থাকে আরও এক ধাপ বাড়িয়ে নেওয়ার সুবিধা দেয়।

১. নেভিগেট করুন **Settings > Security > USB accessories**-এ। ২. সেটিংটিকে **Allow new USB peripherals when unlocked** থেকে পরিবর্তন করে **Deny new USB peripherals** করে দিন।

এই পরিবর্তনটি এক্সিকিউট করার মাধ্যমে, আপনি স্থায়ীভাবে আপনার USB-C পোর্টটিকে একটি গ্লোবাল "Charging-only" স্টেটে নিয়ে গেছেন। এমনকি যদি আপনার ফোন পুরোপুরি আনলক করাও থাকে এবং আপনার হাতে অ্যাক্টিভ অবস্থায় থাকে, তবুও এটি কোনো নতুন USB ডেটা কানেকশন চিনতে অস্বীকার করবে। আপনি সক্রিয়ভাবে ফোনটি ব্যবহার করার সময় যদি কোনো প্রতিপক্ষ আপনার ফোনে একটি USB Rubber Ducky প্লাগ করারও চেষ্টা করে, তবে সেই কিস্ট্রোক ইনজেকশনটি (keystroke injection) ব্যর্থ হবে।

আপনার কেবল তখনই এই ডিফেন্সটি কমানো উচিত (ডিফল্ট সেটিংয়ে ফিরে যাওয়া) যখন কোনো ট্রাস্টেড ল্যাপটপে বা বিশ্বস্ত কম্পিউটারে ফাইল ট্রান্সফার করার সুস্পষ্ট প্রয়োজন দেখা দেয়, এবং ট্রান্সফার শেষ হওয়ামাত্রই আপনার উচিত পুনরায় এই লকডাউনটি চালু করা।

> 💡 ***Pro Tip:*** *আপনার ডিভাইসটিকে কখনোই কোনো পাবলিক চার্জিং স্টেশন, রেন্টাল কার বা ভাড়া করা গাড়ির USB পোর্ট, অথবা হোটেল রুমের USB আউটলেটে প্লাগ করবেন না। একটি ডেডিকেটেড USB ডেটা ব্লকার (USB data blocker) ব্যবহার করুন (এটি একটি ছোট অ্যাডাপ্টার যা ফিজিক্যালি ডেটা পিনগুলোকে ডিসকানেক্ট করে দেয়) অথবা আপনার নিজস্ব AC ওয়াল চার্জার বহন করুন। GrapheneOS-এর চার্জিং-অনলি মোডটি শক্তিশালী সফটওয়্যার প্রোটেকশন বা সুরক্ষা দেয় ঠিকই, কিন্তু একটি হার্ডওয়্যার ডেটা ব্লকার নিশ্চিত নিরাপত্তার একটি ফিজিক্যাল লেয়ার বা স্তর যোগ করে।*

Re-securing the Developer Backdoor (ADB)

পরিশেষে, আমাদের অবশ্যই সবচেয়ে চূড়ান্ত বা আল্টিমেট USB দুর্বলতাটি মোকাবিলা করতে হবে: আর তা হলো **Android Debug Bridge (ADB)**।

ADB হলো ডেভেলপারদের জন্য ডিজাইন করা অত্যন্ত শক্তিশালী একটি কমান্ড-লাইন টুল (command-line tool)। এটি USB-র মাধ্যমে যুক্ত থাকা কোনো কম্পিউটারকে ফোনে ডিপ, সিস্টেম-লেভেলের (system-level) কমান্ড এক্সিকিউট করার, নীরবে অ্যাপ্লিকেশন ইনস্টল করার এবং ডিবাগিং লগ (debugging logs) এক্সট্রাক্ট করার সুবিধা দেয়। যেহেতু ADB সাধারণ বা স্ট্যান্ডার্ড গ্রাফিকাল ইন্টারফেসটি বাইপাস করে সরাসরি সিস্টেম ডিমন (system daemon)-এর সাথে ইন্টার‍্যাক্ট করে, তাই এটি নিরাপত্তার জন্য একটি বিশাল বড় ঝুঁকি।

১.৩ নং সেকশনে, বুটলোডার আনলক (bootloader unlock) করার জন্য আমরা Developer Options চালু করার প্রক্রিয়াটি দেখিয়েছিলাম। একবার সফলভাবে GrapheneOS ইনস্টল করা এবং বুটলোডারটি লক করা হয়ে গেলে, **Developer Options অবশ্যই স্থায়ীভাবে ডিজেবল বা নিষ্ক্রিয় করে দিতে হবে।**

> ⚠ ***Warning:*** *প্রোডাকশনে ব্যবহৃত কোনো ডিভাইসে (production device) Developer Options বা ADB চালু বা এনাবল (enabled) করে রাখাটা ঠিক যেন একটি ব্যাংকের ভল্টে বা সিন্দুকে বড় একটি দরজা লাগিয়ে পেছনের কোনো সার্ভিস হ্যাচ (service hatch) পুরোপুরি খোলা রেখে দেওয়ার সমান। ADB মূলত USB-র মাধ্যমে ফুল সিস্টেম-লেভেল শেল অ্যাক্সেস (shell access) প্রদান করে। ইনস্টলেশনের পরপরই এটি ডিজেবল করে দিন এবং যদি নির্দিষ্ট, সাময়িক কোনো প্রয়োজন না পড়ে তবে কখনোই এটি পুনরায় এনাবল করবেন না।*

১. নেভিগেট করুন **Settings** > **System** > **Developer options**-এ। ২. স্ক্রিনের উপরের মাস্টার সুইচটি টগল করে **Off** করে দিন। ৩. ভেরিফাই বা নিশ্চিত করুন যে Developer Options আর দেখা যাচ্ছে না। যদি আপনি কেবল বিল্ড নম্বরে (build number) ট্যাপ করে এটি এনাবল করার অপশনটি দেখতে পান, তবে তাতে হাত দেবেন না—আপনি ইতিমধ্যেই সুরক্ষিত আছেন।

কোনো প্রোডাকশন ডিভাইসে ADB বা Developer Options চালু রাখা আর ব্যাংকের ভল্টের দরজা লাগিয়ে পেছনের সার্ভিস হ্যাচ খোলা রাখা ঠিক একই বিষয়। ADB ডিজেবল করে এবং GrapheneOS-এর USB-C Port Control কঠোরভাবে প্রয়োগ করার মাধ্যমে, আপনি আপনার সর্বশেষ ফিজিক্যাল দুর্বলতাটিকেও সিল (seal) বা তালাবদ্ধ করে দিলেন। হার্ডওয়্যার লকড (locked)। ডেটা লাইনগুলো বিচ্ছিন্ন (severed)। দুর্গটির নিরাপত্তা এবার সম্পূর্ণ বা কমপ্লিট।

Part II Conclusion: The Fortress Blueprint

চ্যাপ্টার ১-এ আপনি যখন এই যাত্রা শুরু করেছিলেন, তখন আপনার ডিভাইসটি ছিল একটি লায়াবিলিটি বা দুর্বলতার জায়গা। এটি ছিল মূলত একটি মাস-মার্কেট সার্ভেইলেন্স বীকন (mass-market surveillance beacon), যা আপনার ফিজিক্যাল মুভমেন্ট লগ করার জন্য, আপনার ব্যক্তিগত যোগাযোগের সূচি তৈরি করার জন্য এবং কর্পোরেট ডেটা ব্রোকারদের একটি গ্লোবাল নেটওয়ার্কে আপনার ডিজিটাল ফিঙ্গারপ্রিন্ট সম্প্রচার করার জন্য ইঞ্জিনিয়ার করা হয়েছিল। এটি এমনভাবে ডিজাইন করা হয়েছিল যা সম্পূর্ণভাবে আপনার প্রাইভেসির বা গোপনীয়তার বিনিময়ে আপনাকে কেবল সুবিধাই (convenience) প্রদান করত।

Part II চলাকালীন, আপনি অত্যন্ত সুশৃঙ্খলভাবে সেই সার্ভেইলেন্স আর্কিটেকচারটিকে (surveillance architecture) ভেঙে গুঁড়িয়ে দিয়েছেন। আপনি স্রেফ কয়েকটি সেটিংস পরিবর্তন করেননি; আপনি মূলত অপারেটিং সিস্টেমের ইন্টারনাল লজিক বা অভ্যন্তরীণ যুক্তির একটি মৌলিক এবং ক্রিপ্টোগ্রাফিক ওভারহল (cryptographic overhaul) এক্সিকিউট করেছেন। আপনি অন্ধ বিশ্বাসের (implicit trust) জায়গাটি গাণিতিক ভেরিফিকেশন দিয়ে প্রতিস্থাপন করেছেন। আপনি কনভিনিয়েন্স বা সুবিধার জায়গাকে অত্যন্ত সুশৃঙ্খল কম্পার্টমেন্টালাইজেশন (compartmentalization) দিয়ে প্রতিস্থাপন করেছেন।

ডিভাইসটির সাথে আপনার দৈনন্দিন ব্যবহারের ক্ষেত্রে হয়তো আপনি একটি সুস্পষ্ট পরিবর্তন লক্ষ্য করেছেন। আপনাকে এখন অনেক বেশি ঘন ঘন পাসওয়ার্ড এন্টার করতে হয়। আপনার ব্যাংক ব্যালেন্স চেক করার জন্য আপনাকে এখন ইচ্ছাকৃতভাবে প্রোফাইল সুইচ (switch profiles) করতে হয়। আপনাকে এখন সচেতনভাবে আপনার নেটওয়ার্ক কানেকশনগুলো মনিটর করতে হয় এবং অ্যাপ্লিকেশন স্কোপগুলো (application scopes) ম্যানেজ করতে হয়।

এই ফ্রিকশন (friction) বা বাধাকে কোনো বিরক্তি হিসেবে দেখবেন না। এলিট অপারেশনাল সিকিউরিটিতে (operational security), ফ্রিকশন হলো ডিফেন্স মেকানিজম বা প্রতিরক্ষাব্যবস্থা ঠিক যেভাবে কাজ করার কথা, ঠিক সেভাবেই কাজ করার একটি ফিজিক্যাল অনুভূতি। সেই সংক্ষিপ্ত মুহূর্তের অসুবিধাটিই মূলত সেই গাণিতিক বাধা, যা কোনো ক্ষতিকর অ্যাপ্লিকেশনকে নীরবে আপনার ফাইল সিস্টেমের ভেতর দিয়ে ধ্বংসযজ্ঞ চালানো থেকে বিরত রাখে। এটি মূলত সেই ইস্পাতের তৈরি বুল্কহেডেরই (steel bulkhead) ওজন।

আপনি সফলভাবে নির্মাণ করেছেন **The Fortress** বা সেই দুর্ভেদ্য দুর্গ।

The Tactical Recap Checklist

এই কারিকুলামের বা পাঠ্যক্রমের চূড়ান্ত ধাপে যাওয়ার আগে, আপনার বর্তমান ডিফেন্সিভ পসচার (defensive posture) বা প্রতিরক্ষামূলক অবস্থান অডিট করা বা খতিয়ে দেখা অত্যাবশ্যক। একজন এলিট অপারেটর কোনো দুর্বলতা প্রকাশ করেন না। ভেরিফাই বা নিশ্চিত করুন যে আপনার ডিভাইসটি নিচের এই Fortress Blueprint-টি মেনে চলছে:

- **Extreme Compartmentalization:** আপনি "হলো শিপ" (hollow ship) মডেলটি পরিত্যাগ করেছেন। আপনার ডিভাইসটি এখন কঠোরভাবে আইসোলেটেড বা বিচ্ছিন্ন User Profiles-এ বিভক্ত (যেমন, একটি স্টেরাইল (sterile) Owner প্রোফাইল, Signal/ProtonMail-এর জন্য একটি সুরক্ষিত Daily Driver এবং Sandboxed Google Play রান করা একটি কোয়ারেন্টাইনড (quarantined) Banking প্রোফাইল)।
- **Storage and Contact Quarantine:** আপনি এখন আর আপনার আনস্ট্রাকচার্ড ডেটা (unstructured data) বা আপনার সোশ্যাল গ্রাফে ঢালাওভাবে অ্যাক্সেস দেন না। ক্ষতিকর অ্যাপ্লিকেশনগুলোকে গাণিতিকভাবে Storage Scopes এবং Contact Scopes-এর মাধ্যমে রেস্ট্রিক্ট বা সীমাবদ্ধ করা হয়েছে, যা তাদের একটি ফ্যাব্রিকেটেড (fabricated) বা বানোয়াট বাস্তবতা প্রদান করে।
- **The Network Kill Switches:** লোকাল এবং অবিশ্বস্ত অ্যাপ্লিকেশনগুলোর ইন্টারনেট অ্যাক্সেস স্থায়ীভাবে বিচ্ছিন্ন করতে আপনি GrapheneOS-এর নেটওয়ার্ক টগলগুলোকে ওয়েপোনাইজ (weaponized) বা হাতিয়ার হিসেবে ব্যবহার করেছেন, যা তাদের ডেটা এক্সফিলট্রেট (exfiltrate) বা পাচার করার ক্ষমতাকে নিষ্ক্রিয় করে দেয়।
- **The Encrypted Tunnel:** আপনার এক্সটার্নাল বা বাহ্যিক নেটওয়ার্ক ফুটপ্রিন্ট (network footprint) সিল করা আছে। "Always-On" এবং "Block connections without VPN" কিল সুইচ চালু অবস্থায় একটি WireGuard VPN রান করছে। কানেক্টিভিটি চেক (connectivity checks) ডিজেবল বা নিষ্ক্রিয় করা আছে এবং Private DNS (DoT) একটি সুরক্ষিত প্রোভাইডারের মাধ্যমে সমস্ত ডোমেইন কোয়েরি (domain queries) রাউট করছে।
- **Radio Defense:** আপনি LTE-Only মোড চালু করেছেন, যা IMSI Catcher (Stingrays) এবং 2G ডাউনগ্রেড অ্যাটাকের হুমকিকে নিষ্ক্রিয় করে দিয়েছে। Wi-Fi এবং Bluetooth ব্যাকগ্রাউন্ড স্ক্যানিং ডিজেবল করা আছে এবং Per-Connection MAC Randomization আপনার ফিজিক্যাল মুভমেন্ট বা চলাফেরাকে কর্পোরেট হিটম্যাপ (corporate heatmaps) থেকে রক্ষা করছে।
- **The Deadman's Switch:** আপনার Auto-Reboot টাইমারটি আপনার থ্রেট মডেল (যেমন, ৪ ঘণ্টা) অনুযায়ী কঠোরভাবে ক্যালিব্রেট (calibrated) বা সেট করা আছে, যা নিশ্চিত করে যে ডিভাইসটি ফিজিক্যালি বাজেয়াপ্ত হলে এটি এর RAM মুছে ফেলবে এবং একটি অভেদ্য BFU স্টেটে ফিরে যাবে।
- **The Nuclear Option:** একটি Duress PIN কনফিগার করা আছে, যা চরম ফিজিক্যাল জোর-জবরদস্তির বা শারীরিক বলপ্রয়োগের ক্ষেত্রে একটি স্থায়ী এবং অপরিবর্তনীয় ক্রিপ্টোগ্রাফিক ওয়াইপ মেকানিজম (cryptographic wipe mechanism) প্রদান করে।

- **Physical Port Lockdown:** USB-C পোর্টটি "Charging-only when locked"-এ সেট করা আছে, যা স্ক্রিন অন্ধকার হওয়ামাত্রই ফিজিক্যালি ডেটা লাইনগুলোকে বিচ্ছিন্ন করে দেয় এবং Cellebrite-এর মতো ফরেনসিক এক্সট্রাকশন টুলগুলোকে একেবারে অকেজো করে দেয়। Developer Options (ADB) স্থায়ীভাবে ডিজেবল করা আছে।

The Cost of Attack

এই ব্লুপ্রিন্টটি (blueprint) এক্সিকিউট করার মাধ্যমে, আপনি মূলত আপনার ডিজিটাল আইডেন্টিটি বা পরিচয়কে কেন্দ্র করে সাইবার ওয়ারফেয়ারের (cyber warfare) পুরো অর্থনীতিকেই মৌলিকভাবে পরিবর্তন করে দিয়েছেন।

আপনি আর ড্র্যাগনেট সার্ভেইলেন্স (dragnet surveillance), প্যাসিভ ডেটা হার্ভেস্টিং (passive data harvesting), অথবা স্বয়ংক্রিয় ম্যালওয়্যার স্ক্রিপ্টের কাছে দুর্বল বা সংবেদনশীল নন। চরম পর্যায় পর্যন্ত হার্ডেনড বা সুরক্ষিত করা এমন একটি ডিভাইসকে সফলভাবে কম্প্রোমাইজ করতে হলে, কোনো প্রতিপক্ষ আর সস্তা চালের ওপর নির্ভর করতে পারে না। তাদেরকে অবশ্যই অত্যন্ত আধুনিক, বহু-মিলিয়ন ডলার মূল্যের জিরো-ডে এক্সপ্লয়েট চেইন (zero-day exploit chains) ডিপ্লয় করতে হবে। তাদের অবশ্যই রাষ্ট্র-পর্যায়ের (state-level) সক্ষমতা থাকতে হবে। আপনি মূলত আপনার ওপর আক্রমণ করার খরচকে শূন্য থেকে বাড়িয়ে অ্যাস্ট্রোনমিক্যাল (astronomical) বা আকাশচুম্বী করে তুলেছেন।

Transitioning to Part III: The Battlefield

Part I আপনাকে শিখিয়েছে কীভাবে হার্ডওয়্যার ইনস্টল করতে হয়। Part II আপনাকে শিখিয়েছে কীভাবে প্রতিরক্ষাব্যবস্থা বা ডিফেন্স আর্কিটেক্ট (architect) করতে হয়।

বুল্কহেডগুলো (bulkheads) ঠিক *কীভাবে* কাজ করে, তা আপনি জানেন, কিন্তু একটি দুর্গের প্রকৃত পরীক্ষা তখনই হয় যখন এটি অবরুদ্ধ (under siege) থাকে। এই বইয়ের চূড়ান্ত পর্বে, আমরা থিওরি বা তত্ত্বের গণ্ডি পেরিয়ে অ্যাক্টিভ থ্রেট ল্যান্ডস্কেপে (active threat landscape) প্রবেশ করব।

আমরা এগিয়ে যাচ্ছি **Part III: The Battlefield**-এর দিকে।

আগামী চ্যাপ্টারগুলোতে, আমরা আপনার ডিভাইসের বিরুদ্ধে রিয়েল-ওয়ার্ল্ড (real-world), এলিট সাইবার অ্যাটাকগুলো (cyberattacks) সিমুলেট বা অনুকরণ করব। আমরা ব্যবচ্ছেদ করে দেখাব যে কীভাবে রাষ্ট্র-সমর্থিত হ্যাকাররা বা নেশন-স্টেট অ্যাক্টররা (nation-state actors) মিলিয়ন-ডলার মূল্যের "Zero-Click" স্পাইওয়্যার (যেমন Pegasus) ডিপ্লয় করে। আপনার Sandboxed Google Play লেয়ারকে বাইপাস করার জন্য ডিজাইন করা অত্যাধুনিক এবং টার্গেটেড স্পিয়ার-

ফিশিং (spear-phishing) ক্যাম্পেইনগুলো আমরা বিশ্লেষণ করব। আমরা আলোচনা করব কোনো বৈরী আন্তর্জাতিক বর্ডারে একটি চরম আক্রমণাত্মক ইলেক্ট্রনিক সার্চ বা তল্লাশির হাত থেকে বেঁচে ফেরার একদম নিখুঁত, মিনিট-বাই-মিনিট (minute-by-minute) প্রোটোকল নিয়ে।

আমরা আপনাকে ঠিক সেটাই দেখাব যে ঠিক *কেন* Part II-এর এই ফ্রিকশন বা বাধাগুলো এত প্রয়োজনীয়, এবং আমরা এটি প্রমাণ করে দেখাব যে কীভাবে GrapheneOS আর্কিটেকচার এই গ্রহের সবচেয়ে ভয়ংকর ডিজিটাল অস্ত্রগুলোকে হজম করে (absorbs), পথভ্রষ্ট করে (deflects) এবং নিষ্ক্রিয় (neutralizes) করে দেয়।

আপনার আর্মার (armor) বা বর্ম এখন তৈরি। আঘাতের (impact) জন্য প্রস্তুত হোন।

Part III: The Battlefield

Chapter 3 Introduction: Live-Fire Environment

আপনার ট্রেনিংয়ের থিওরেটিক্যাল বা তাত্ত্বিক পর্বটি শেষ হয়েছে।

Part I-এ, আমরা ক্রিপ্টোগ্রাফিক বেইজলাইন (cryptographic baseline) প্রতিষ্ঠা করেছি এবং হার্ডওয়্যারকে সুরক্ষিত করেছি। Part II-এ, আমরা ইন্টারনাল বুল্কহেডগুলোর (internal bulkheads) কাঠামো তৈরি করেছি, কঠোর কম্পার্টমেন্টালাইজেশন (compartmentalization) এক্সিকিউট করেছি এবং অননুমোদিত নেটওয়ার্কগুলোকে বিচ্ছিন্ন করেছি। এই ব্লুপ্রিন্টটি সঠিকভাবে অনুসরণ করলে আপনার ডিভাইসটি আর কোনো সাধারণ স্মার্টফোন থাকবে না। এটি পরিণত হবে গাণিতিকভাবে ভেরিফায়েড, মিলিটারী-গ্রেড (military-grade) একটি ভল্টে।

কিন্তু পরীক্ষা না করা পর্যন্ত একটি ভল্ট মূলত নিছকই একটি হাইপোথিসিস (hypothesis) বা অনুমানের চেয়ে বেশি কিছু নয়।

এলিট সাইবার সিকিউরিটিতে কোনো থিওরি বা তত্ত্বের কোনো অর্থ নেই, যতক্ষণ না তা প্রতিপক্ষের সাথে লড়াই করে টিকে থাকতে পারে। থ্রেট ল্যান্ডস্কেপ (threat landscape) মানে শুধু বিজ্ঞাপন বিক্রি করা কিছু অ্যালগরিদম নয়। এখানে রাজত্ব করে Advanced Persistent Threats (APTs), রাষ্ট্র-সমর্থিত গোয়েন্দা বাহিনী (nation-state intelligence), সাইবার অপরাধীদের সিন্ডিকেট এবং NSO Group-এর মতো প্রাইভেট ইন্টেলিজেন্স কন্ট্রাক্টররা।

এই গোষ্ঠীগুলো কোনো ইউজার অ্যাগ্রিমেন্ট বা ব্যবহারকারী চুক্তির তোয়াক্কা করে না। আপনার মাইক্রোফোনে অ্যাক্সেস নেওয়ার জন্য তারা আপনার অনুমতি চায় না। তাদের কাছে রয়েছে বহু-মিলিয়ন ডলারের বিশাল বাজেট, জিরো-ডে এক্সপ্লয়েটের (zero-day exploit) এক বিশাল অস্ত্রাগার এবং টার্গেটের ডিভাইস একবারও স্পর্শ না করেই সাধারণ কমার্শিয়াল অপারেটিং সিস্টেমকে পুরোপুরি ধ্বংস করে দেওয়ার অসীম টেকনিক্যাল সক্ষমতা।

আপনার দৈনন্দিন ডিজিটাল জীবনে আপনি যে ফ্রিকশন (friction) বা বাধা তৈরি করেছেন, তার প্রকৃত মূল্য বুঝতে হলে আপনাকে অবশ্যই স্বচক্ষে দেখতে হবে যে কীভাবে এই ডিফেন্সগুলো অ্যাক্টিভলি বা সক্রিয়ভাবে এই থ্রেটগুলোকে প্রতিহত করে।

আর এটিই আমাদেরকে Part III-এর দিকে নিয়ে যায়। আমরা এখন একটি সাধারণ ইন্সট্রাকশনাল ম্যানুয়াল (instructional manual) থেকে একটি অ্যাডভান্সড ফরেনসিক সিকিউরিটি ব্রিফিংয়ে (advanced forensic security briefing) স্থানান্তরিত হচ্ছি। আমরা এখন রণাঙ্গনে বা যুদ্ধক্ষেত্রে (battlefield) পা রাখতে যাচ্ছি।

৩.০ The Rules of Engagement

পরবর্তী চ্যাপ্টারগুলোতে, আমরা উচ্চ-ক্ষমতাসম্পন্ন বা হাই-ফিডেলিটি (high-fidelity) থ্রেট সিমুলেশনের (threat simulations) একটি সিরিজ এক্সিকিউট করব। আমরা একজন নিষ্ক্রিয় ইউজারের (passive user) দৃষ্টিভঙ্গি ত্যাগ করব এবং এলিট পেনিট্রেশন টেস্টার (penetration testers) ও ইন্টেলিজেন্স অ্যানালিস্টদের (intelligence analysts) ব্যবহার করা অ্যানালিটিকাল ফ্রেমওয়ার্ক বা বিশ্লেষণাত্মক কাঠামোটি গ্রহণ করব: **Red Team versus Blue Team**।

The Red Team (The Adversary)

Red Team মূলত থ্রেট অ্যাক্টর (threat actor) বা প্রতিপক্ষের প্রতিনিধিত্ব করে। তাদের একমাত্র উদ্দেশ্য হলো ডিভাইসের নিরাপত্তা ব্রিচ (breach) করা বা ভাঙা, রুট (root) লেভেলে ক্ষমতা বৃদ্ধি করা বা এস্কেলেট (escalate) করা এবং টার্গেটের ইন্টেলিজেন্স এক্সফিলট্রেট (exfiltrate) বা চুরি করা—অথবা স্থায়ী কোনো স্পাইওয়্যার ডিপ্লয় করা।

এই পুরো সেকশন জুড়ে, বর্তমানে বাস্তবে (in the wild) ব্যবহৃত সবচেয়ে ধ্বংসাত্মক ডিজিটাল এবং ফিজিক্যাল অস্ত্রগুলোতে Red Team-কে অ্যাক্সেস দেওয়া হবে। আমরা যেসব অস্ত্র ব্যবহার করে অ্যাটাকগুলো সিমুলেট করব:

- **Zero-Click Remote Code Execution (RCE) Exploits:** মিলিটারী-গ্রেড স্পাইওয়্যারের (যেমন Pegasus) ডিপ্লয়মেন্ট, যা কোনো স্ট্যান্ডার্ড ডিভাইসকে কম্প্রোমাইজ করার জন্য ইউজারের সাথে বিন্দুমাত্র কোনো ইন্টারঅ্যাকশনেরও প্রয়োজন বোধ করে না।
- **Physical Forensic Extraction:** বর্ডার কন্ট্রোল এবং আইন প্রয়োগকারী সংস্থার দ্বারা ব্যবহৃত বিশেষায়িত হার্ডওয়্যারের (যেমন Cellebrite বা GrayKey টার্মিনাল) ডিপ্লয়মেন্ট, যা লক স্ক্রিন ব্রুট-ফোর্স (brute-force) করতে এবং RAM স্টেট (RAM states) এক্সট্রাক্ট করতে ব্যবহৃত হয়।
- **Targeted Spear-Phishing & Social Engineering:** অত্যন্ত কাস্টমাইজড এবং মনস্তাত্ত্বিকভাবে ম্যানিপুলেটিভ বা বিভ্রান্তিকর ফাঁদ, যা হিউম্যান অপারেটরকে ধোঁকা দিয়ে কি (keys) সারেন্ডার করতে বাধ্য করার মাধ্যমে হার্ডওয়্যার সিকিউরিটি বাইপাস করার জন্য ডিজাইন করা।
- **Baseband and Network Exploitation:** Man-in-the-Middle (MitM) এবং ডাউনগ্রেড অ্যাটাক এক্সিকিউট করার জন্য IMSI ক্যাচার (Stingrays) এবং ম্যালিসিয়াস Wi-Fi অ্যাক্সেস পয়েন্টের ডিপ্লয়মেন্ট।

The Blue Team (The Defender)

Blue Team আপনার অর্থাৎ অপারেটরের প্রতিনিধিত্ব করে, যে শুধুমাত্র Fortress Blueprint (Part II) অনুযায়ী কনফিগার করা GrapheneOS আর্কিটেকচার দিয়ে সজ্জিত।

Blue Team-এর উদ্দেশ্য হলো গাণিতিকভাবে Red Team-কে তাদের লক্ষ্য অর্জনে বাধা দেওয়া বা ডিনাই (deny) করা। আমরা কোনোভাগ্যের ওপর নির্ভর করব না, এবং আমরা প্রতিপক্ষের কোনো ভুল করার অপেক্ষায় বসে থাকব না। আমরা সম্পূর্ণভাবে অপারেটিং সিস্টেমের ক্রিপ্টোগ্রাফিক এবং আর্কিটেকচারাল সুপিরিওরিটির (architectural superiority) বা শ্রেষ্ঠত্বের ওপর নির্ভর করব।

Analyzing the Kill Chain

সাইবার সিকিউরিটিতে, একটি সফল ব্রিচ (breach) খুব কম সময়ই জাদুকরী কোনো একক কোডের লাইনের মাধ্যমে অর্জন করা যায়। এর জন্য প্রয়োজন হয় একটি "Exploit Kill Chain"—যা মূলত বিভিন্ন ঘটনার একটি অত্যন্ত সুনির্দিষ্ট সিকোয়েন্স।

স্পাইওয়্যার ডিপ্লয় করার জন্য, প্রতিপক্ষকে অবশ্যই ইনিশিয়াল বা প্রাথমিক অ্যাক্সেসের জন্য কোনো ব্রাউজার ভালনারেবিলিটি বা দুর্বলতা (Step 1) খুঁজে বের করতে হয়। প্রিভিলেজ এস্কেলেট বা সুযোগ-সুবিধা বৃদ্ধি করার জন্য মেমোরি অ্যালোকেটরে (memory allocator) তাদের একটি দ্বিতীয় দুর্বলতা (Step 2) খুঁজতে হয়। সবশেষে, স্যান্ডবক্স ভেঙে মূল কোর ওএস-এ (core OS) অ্যাক্সেস নেওয়ার জন্য তাদের একটি তৃতীয় দুর্বলতা (Step 3) খুঁজে বের করতে হয়। এই চেইনের *যেকোনো* একটি ধাপ বা স্টেপ ভেঙে দিন, আর সাথে সাথেই তাদের বহু-মিলিয়ন ডলারের পুরো অ্যাটাকটিই সম্পূর্ণ ব্যর্থ হয়ে যাবে।

এই সেকশনে, আমরা ফরেনসিকভাবে বা অত্যন্ত নিখুঁতভাবে এই কিল চেইনগুলোকে (kill chains) ব্যবচ্ছেদ করব।

আমরা শুধু এইটুকু বলেই শেষ করব না যে "GrapheneOS অনেক সুরক্ষিত।" রিয়েল-টাইমে একটি আক্রমণ ঘটার সাথে সাথেই আমরা সেই আক্রমণের অ্যানাটমি (anatomy) বা গঠনশৈলী বিশ্লেষণ করব। আমরা দেখব কীভাবে Red Team একটি মেমোরি করাপশন এক্সপ্লয়েট (memory corruption exploit) লঞ্চ করে, এবং আমরা খুব নিখুঁতভাবে দেখব যে কীভাবে GrapheneOS-এর `hardened_malloc` সেই বাফার ওভারফ্লো (buffer overflow) ডিটেক্ট বা শনাক্ত করে এবং তাৎক্ষণিকভাবে প্রসেসটিকে টার্মিনেট করে Step 2-তে এসেই কিল চেইনটিকে ভেঙে দেয়। আমরা দেখব কীভাবে Red Team একটি URL ইনটেন্টকে (URL intent) হাইজ্যাক করার চেষ্টা করে, এবং আমরা দেখব কীভাবে অপারেটিং সিস্টেম অত্যন্ত কঠোরভাবে `assetlinks.json` ক্রিপ্টোগ্রাফিক সিগনেচার ভেরিফাই বা যাচাই করে Step 1-এই পুরো চেইনটিকে ভেঙে দেয়।

Part III-এর শেষে এসে, আপনি শুধুমাত্র এটাই জানবেন না যে *কীভাবে* GrapheneOS ব্যবহার করতে হয়। এর পেছনের গণিতটি ঠিক *কেন* এত অভেদ্য, তার একটি গভীর, ফরেনসিক বোঝাপড়াও আপনার তৈরি হবে। আপনি আপনার এই আর্মার বা বর্মটির নিখুঁত মেকানিজম বা কাজ করার পদ্ধতিটি বুঝতে সক্ষম হবেন।

সিমুলেশনগুলো (simulations) লোড করা হয়েছে। Red Team-কে এনগেজ (engage) বা আক্রমণ শুরু করার অথরাইজেশন দেওয়া হয়েছে।

স্বাগতম এই যুদ্ধক্ষেত্রে।

৩.১ Understanding the Attack Surface

Red Team কোনো আক্রমণ বা অ্যাসাল্ট (assault) শুরু করার আগে, তাদের অবশ্যই রিকনেসান্স (reconnaissance) বা অনুসন্ধান চালাতে হয়। তাদেরকে অবশ্যই দুর্গের পরিধি বা পেরিমিটার (perimeter) স্ক্যান করতে হয় এবং প্রতিটি সম্ভাব্য দরজা, জানালা এবং ভেন্টিলেশন শ্যাফট (ventilation shaft) চিহ্নিত করতে হয়। সাইবার সিকিউরিটিতে, এই সমস্ত সম্ভাব্য এন্ট্রি পয়েন্টের (entry points) সমষ্টিগত রূপটিকেই **Attack Surface** বা অ্যাটাক সারফেস বলা হয়।

কোনো অপারেটিং সিস্টেমকে সফলভাবে ডিফেন্ড (defend) বা রক্ষা করতে হলে, Blue Team-কে অবশ্যই এই সারফেস বা পৃষ্ঠতল সম্পর্কে একটি নিবিড় বোঝাপড়া থাকতে হবে। তাদের ঠিকভাবে জানতে হবে যে আর্মার বা বর্মটি কোথায় সবচেয়ে বেশি পুরু, আর কোথায় এটি স্বভাবগতভাবেই পাতলা।

The Anatomy of the Mobile Attack Surface

একটি আধুনিক স্মার্টফোন কোডের কোনো একক, ঐক্যবদ্ধ ব্লক (unified block) নয়। এটি হলো হার্ডওয়্যার ড্রাইভার, রেডিও প্রোটোকল, অ্যাপ্লিকেশন প্রোগ্রামিং ইন্টারফেস (APIs) এবং রেন্ডারিং ইঞ্জিনের একটি অবিশ্বাস্য রকমের জটিল এবং ইন্টারকানেক্টেড বা পরস্পর সংযুক্ত ওয়েব বা জাল।

কোনো মোবাইল ডিভাইসের প্রধান অ্যাটাক সারফেসকে মূলত পাঁচটি সুনির্দিষ্ট ভেক্টরে (vectors) ম্যাপ করা যেতে পারে:

১. The Baseband Processor: ২.১৩ নং সেকশনে বিস্তারিত বলা হয়েছে যে, বেসব্যান্ড হলো একটি মাইক্রো-কম্পিউটার যা 2G, 3G, 4G এবং 5G প্রোটোকলের মাধ্যমে সেলুলার টাওয়ারের সাথে নেগোশিয়েট (negotiating) করার জন্য দায়িত্বপ্রাপ্ত। যেহেতু এটি বাতাসের মধ্য দিয়ে আনঅথেনটিকেটেড (unauthenticated) বা যাচাই না করা রেডিও সিগন্যাল প্রসেস করে, তাই এটি রিমোট এক্সপ্লয়েটেশনের (remote exploitation) জন্য অত্যন্ত মূল্যবান একটি লক্ষ্যবস্তু। **২. Local Radio Frequencies:** Wi-Fi, Bluetooth এবং NFC (Near Field Communication) হার্ডওয়্যার স্ট্যাক। এই প্রোটোকলগুলো কাজ করার জন্য ডিভাইসটিকে সক্রিয়ভাবে আশেপাশে থাকা অবিশ্বস্ত বা আনট্রাস্টেড (untrusted) ডিভাইসগুলো থেকে পাঠানো জটিল ডেটা প্যাকেটগুলো শুনতে এবং প্রসেস করতে হয়। **৩. The Physical Data Stack:** USB-C পোর্ট এবং এর সাথে যুক্ত সফটওয়্যার ড্রাইভার। এটি মূলত সেই ভেক্টর (vector) যা ম্যালিসিয়াস বা ক্ষতিকর চার্জিং কিয়স্ক এবং ফরেনসিক এক্সট্রাকশন মেশিনগুলো ব্যবহার করে থাকে। **৪. The Browser Rendering Engine:** ওয়েব ব্রাউজার (Vanadium) হলো ডিভাইসের সবচেয়ে বড় এবং তর্কসাপেক্ষে সবচেয়ে বিপজ্জনক অ্যাটাক সারফেস। এটি মূলত ইন্টারনেটের জুড়ে থাকা র‍্যান্ডম বা যেকোনো সার্ভার থেকে আনট্রাস্টেড কোড (JavaScript, WebAssembly) ডাউনলোড, কম্পাইল এবং এক্সিকিউট করার জন্যই বিশেষভাবে ডিজাইন করা হয়েছে। **৫. System APIs and IPC:** ইন্টারনাল

কমিউনিকেশন চ্যানেল বা অভ্যন্তরীণ যোগাযোগের মাধ্যমগুলো (যেমন Android Intents), যা অ্যাপ্লিকেশনগুলোকে একে অপরের সাথে কথা বলার সুযোগ দেয় এবং ফাইল সিস্টেম বা মাইক্রোফোনে অ্যাক্সেস নেওয়ার জন্য রিকোয়েস্ট (request) করতে দেয়।

যখন কোনো এলিট প্রতিপক্ষ একটি টার্গেটকে ব্রিচ (breach) করার সিদ্ধান্ত নেয়, তখন তারা ডিভাইসে র‍্যান্ডমলি বা এলোমেলোভাবে অ্যাটাক করে না। তারা এই পাঁচটি ভেক্টর বিশ্লেষণ করে, এবং কোডের ভেতরে থাকা একটি সুনির্দিষ্ট ত্রুটি—একটি গাণিতিক ভুল (mathematical error)—খুঁজে বের করার চেষ্টা করে যা তারা এক্সপ্লয়েট করতে পারে।

The Reactionary Model (The CVE Hamster Wheel)

স্ট্যান্ডার্ড কমার্শিয়াল সফটওয়্যার ইন্ডাস্ট্রি (commercial software industry) মূলত একটি রিঅ্যাকশনারি বা প্রতিক্রিয়াশীল (reactionary) সিকিউরিটি মডেলের ওপর ভিত্তি করে কাজ করে।

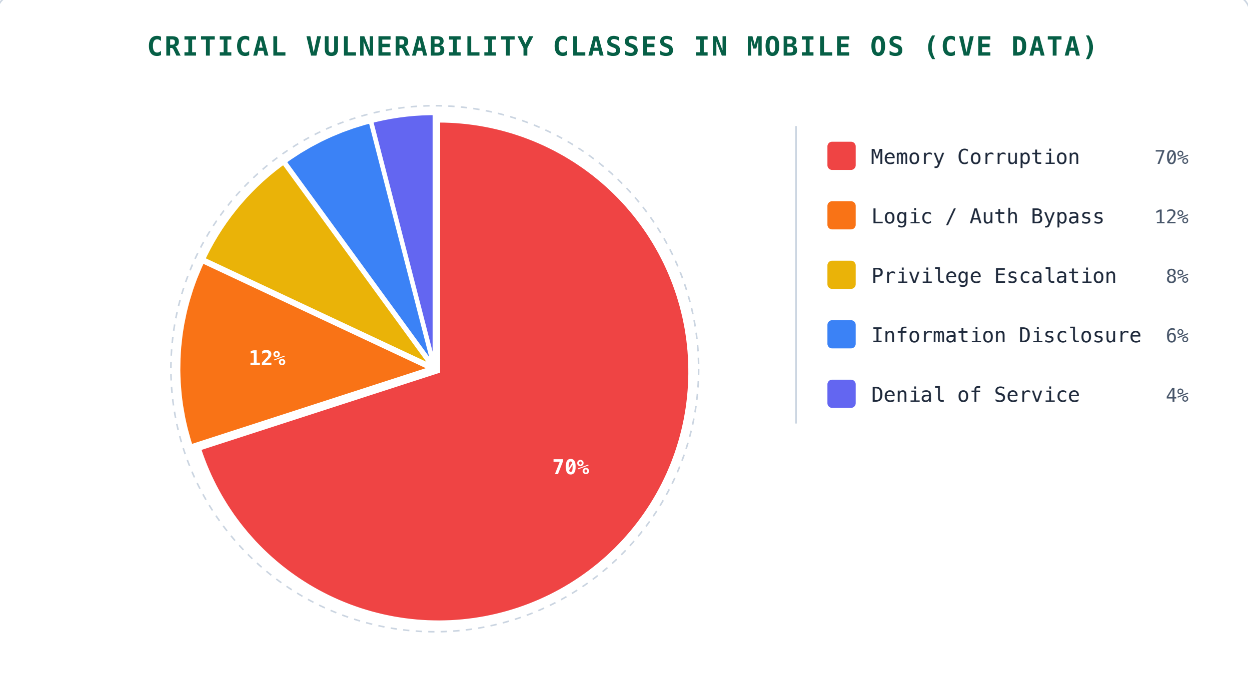
যখন কোনো সিকিউরিটি গবেষক (অথবা কোনো শত্রু গোয়েন্দা সংস্থা) Bluetooth স্ট্যাকে কোনো ভালনারেবিলিটি বা দুর্বলতা আবিষ্কার করেন, তখন সেই দুর্বলতাটি শেষ পর্যন্ত আইডেন্টিফাই করা হয়, তাকে একটি **CVE** (Common Vulnerabilities and Exposures) ট্র্যাকিং নম্বর (tracking number) দেওয়া হয়, এবং ম্যানুফ্যাকচারার বা প্রস্তুতকারকের কাছে রিপোর্ট করা হয়।

প্রস্তুতকারকের ইঞ্জিনিয়াররা সেই CVE-টি বিশ্লেষণ করেন, ওই নির্দিষ্ট গাণিতিক ভুলটি ফিক্স (fix) করার জন্য একটি সফটওয়্যার প্যাচ (software patch) লেখেন, এবং আপনার ফোনে একটি আপডেট পুশ (push) করে পাঠিয়ে দেন। এই প্রক্রিয়াটিতে প্রায়শই কয়েক সপ্তাহ, এবং কখনো কখনো কয়েক মাস পর্যন্ত সময় লেগে যায়।

দুর্বলতা আবিষ্কার হওয়া এবং প্যাচ ইনস্টল হওয়ার মধ্যবর্তী এই সময়টুকুতে—এই দুর্বলতাটি একটি **Zero-Day** হিসেবে পরিচিত থাকে।

Blue Team যদি শুধুমাত্র সিকিউরিটি প্যাচ ইনস্টল করার ওপর নির্ভর করে, তবে তারা এমন একটি ইঁদুর দৌড়ে (unwinnable race) আটকা পড়বে যা কখনো জেতা সম্ভব নয়। তারা সর্বদা প্রতিপক্ষের চেয়ে এক ধাপ পিছিয়ে থাকবে। কোনো রাষ্ট্র-সমর্থিত হ্যাকার (nation-state actor) যদি স্ট্যান্ডার্ড Android ডিভাইসের বিরুদ্ধে কোনো জিরো-ডে এক্সপ্লয়েট (Zero-Day exploit) ব্যবহার করে, তবে ডিভাইসটির কাছে আত্মরক্ষার কোনো উপায়ই থাকে না। কারণ, অপারেটিং সিস্টেম নিজে এখনো জানেই না যে এর ভেতর কী ভেঙে আছে বা নষ্ট হয়ে আছে, তাই এটি কোনোভাবেই প্যাচ তৈরি করতে পারে না।

The GrapheneOS Philosophy: Attack Surface Reduction



Source: Google Project Zero, Microsoft MSRC, and Android Security Bulletins — Memory corruption is the dominant class

GrapheneOS মূলত এই রিঅ্যাকশনারি প্যাচিং মডেলকে (reactionary patching model) পুরোপুরি প্রত্যাখ্যান করে। এই প্রোজেক্টটির কোর বা মূল ইঞ্জিনিয়ারিং ফিলোসফি হলো অত্যন্ত সাধারণ এবং আপোসহীন (uncompromising): **Patching is a reaction; Attack Surface Reduction is a defense.**

কোনো দুর্বলতা বা ভালনারেবিলিটির বিরুদ্ধে নিজেকে রক্ষা করার সবচেয়ে গাণিতিক ও নিখুঁত উপায় হলো এটি নিশ্চিত করা যে, সেই দুর্বল কোডটির যেন শুরুতেই কোনো অস্তিত্ব না থাকে।

আপনি যদি Part II-এ আমাদের তৈরি করা Fortress Blueprint-টির দিকে ফিরে তাকান, তবে দেখবেন যে আমাদের নেওয়া প্রতিটি পদক্ষেপই মূলত Attack Surface Reduction-এরই একটি এক্সারসাইজ বা অনুশীলন ছিল:

- আমরা LTE-Only মোডের মাধ্যমে 2G এবং 3G ডিজেবল করেছিলাম। আমরা 2G-র কোনো দুর্বলতার প্যাচ আসার জন্য অপেক্ষা করে বসে থাকিনি; আমরা স্রেফ গাণিতিকভাবে 2G-র অ্যাটাক সারফেসটিকেই মুছে ফেলেছিলাম।
- আমরা USB-C পোর্টটিকে "Charging-only when locked" অবস্থায় কনফিগার করেছিলাম। আমরা USB ড্রাইভারগুলোর একেবারে নিখুঁত কোডিংয়ের ওপর নির্ভর করিনি; আমরা ফিজিক্যালি ডেটা লাইনগুলোকেই বিচ্ছিন্ন

করে দিয়েছিলাম।

- আমরা Bluetooth ব্যাকগ্রাউন্ড স্ক্যানিং বন্ধ করে দিয়েছিলাম। রেডিও যদি বন্ধ থাকে এবং কিছুই শুনতে অস্বীকৃতি জানায়, তবে ক্ষতিকর কোনো Bluetooth প্যাকেট দিয়ে আমাদেরকে এক্সপ্লয়েট করা অসম্ভব।

অপ্রয়োজনীয় কোডগুলো বাদ দিয়ে, লিগ্যাসি প্রোটোকলগুলো ডিজেবল করে এবং কঠোর কম্পার্টমেন্টালাইজেশন বা বিভাজন প্রয়োগ করে, GrapheneOS ফিজিক্যালি টার্গেটের আকারকে সংকুচিত বা ছোট করে আনে।

Killing Entire Vulnerability Classes

তবে, GrapheneOS-এর প্রকৃত জিনিয়াস বা প্রতিভা লুকিয়ে আছে এটি কীভাবে সেইসব অ্যাটাক সারফেসকে হ্যান্ডেল করে যা কোনোভাবেই রিমুভ (remove) বা মুছে ফেলা *সম্ভব নয়*। আপনি চাইলেই স্রেফ মেমোরি অ্যালোকেটর (memory allocator) ডিলিট করে দিতে পারেন না, এবং ওয়েব ব্রাউজার ডিলিট করতে গেলে আপনার ডিভাইসটি পুরোপুরি অকেজো হয়ে পড়বে।

ইন্ডিভিজুয়াল বা আলাদা আলাদা CVE প্যাচ করার অপেক্ষায় বসে থাকার বদলে, GrapheneOS **Exploit Mitigations** ডিপ্লয় করে। এগুলো হলো মূলত আর্কিটেকচারাল (architectural) বা কাঠামোগত পরিবর্তন, যা একযোগে দুর্বলতার পুরো *ক্লাস* (classes) বা ধরনকেই নিষ্ক্রিয় করার জন্য ডিজাইন করা।

Memory Corruption-এর বিষয়টি বিবেচনা করুন।

আধুনিক অপারেটিং সিস্টেমগুলোতে থাকা সমস্ত মারাত্মক দুর্বলতার প্রায় ৭০% হলো মেমোরি করাপশন বাগ (memory corruption bugs) (যেমন বাফার ওভারফ্লো বা ইউজ-আফটার-ফ্রি এরর)। একজন আক্রমণকারী কোনো অ্যাপ্লিকেশন কীভাবে তার RAM ম্যানেজ করে তার মধ্যে একটি ত্রুটি খুঁজে পায়, বাফারটি ওভারলোড করে এবং সেই ওভারফ্লোটিকে কাজে লাগিয়ে সিস্টেম মেমোরিতে তাদের নিজস্ব ম্যালিসিয়াস পেলোডটি (malicious payload) ইনজেক্ট করে দেয়।

স্ট্যান্ডার্ড Android এই আলাদা আলাদা বাফার ওভারফ্লো (buffer overflows) বা ত্রুটিগুলো একে একে খুঁজে বের করার এবং প্যাচ করার চেষ্টা করে। GrapheneOS এই সম্পূর্ণ ক্লাসটিকে বা ধরনটিকে একদম গোড়া থেকে মোকাবিলা করে।

GrapheneOS স্ট্যান্ডার্ড Android মেমোরি অ্যালোকেটরকে `hardened_malloc` দিয়ে প্রতিস্থাপন করে। শুধু তাই নয়, আধুনিক Pixel হার্ডওয়্যারে এটি অত্যন্ত ব্যাপকভাবে ARM Memory Tagging Extension (MTE)-এর সুবিধা কাজে লাগায়।

ব্রাউজারে কোনো সুনির্দিষ্ট জিরো-ডে (Zero-Day) CVE আছে কি না, তা নিয়ে এই মিটিগেশনগুলোর (mitigations) কোনো মাথাব্যথা নেই। বরং, `hardened_malloc` এবং MTE ফান্ডামেন্টাল বা মৌলিকভাবেই সিস্টেম মেমোরির পদার্থবিদ্যাকে বা ফিজিক্সকে পরিবর্তন করে দেয়। এগুলো অত্যন্ত অ্যাগ্রেসিভ বা আক্রমণাত্মক র‍্যান্ডমাইজেশন (randomization) এবং কঠোর বাউন্ডারি চেকিং (strict boundary checking) বা সীমানা যাচাইকরণ ব্যবস্থার প্রবর্তন করে।

Red Team যখন Vanadium-এ বাফার ওভারফ্লো (buffer overflow) ট্রিগার করার জন্য একটি বহু-মিলিয়ন ডলার মূল্যের জিরো-ডে এক্সপ্লয়েট লঞ্চ করে, তখন এক্সপ্লয়েটটি ঠিক সেভাবেই ফায়ার করে বা কাজ করে যেভাবে এটি ডিজাইন করা হয়েছে। কিন্তু, অত্যন্ত ক্লিন বা নিখুঁতভাবে পেলোডটি ইনজেক্ট করার বদলে, এক্সপ্লয়েটটি সোজা গিয়ে ধাক্কা খায় সেই হার্ডেনড মেমোরি বাউন্ডারিতে (hardened memory boundary)। `hardened_malloc` তাৎক্ষণিকভাবে গাণিতিক এই অসঙ্গতিটি বা অ্যানোমালিটি (anomaly) শনাক্ত করে, ধরে নেয় যে সিস্টেমটি অ্যাটাকের বা আক্রমণের শিকার হচ্ছে, এবং ম্যালিসিয়াস পেলোডটি এক্সিকিউট হওয়ার আগেই জোরপূর্বক অ্যাপ্লিকেশন প্রসেসটিকে ক্র্যাশ (crashes) করে দেয়।

এক্সপ্লয়েটটি ব্যর্থ হয়। টার্গেট সুরক্ষিত থাকে। এবং সবচেয়ে গুরুত্বপূর্ণ বিষয় হলো, GrapheneOS অ্যাটাকটিকে নিষ্ক্রিয় করে দেয় *এমনকি এটি না জেনেই যে ওই নির্দিষ্ট দুর্বলতাটির আদৌ কোনো অস্তিত্ব ছিল*।

Attack Surface Reduction এবং অত্যন্ত অ্যাগ্রেসিভ Exploit Mitigations-এর ওপর ফোকাস করার মাধ্যমে, GrapheneOS এই রিঅ্যাকশনারি সাইকেল (reactionary cycle) বা প্রতিক্রিয়াশীল চক্রটি ভেঙে ফেলে। এটি শুধু যে উঁচু দেয়াল তৈরি করে তা নয়; এটি যুদ্ধক্ষেত্রের পুরো ফিজিক্সকেই (physics) বা পদার্থবিদ্যাকেই বদলে দেয়, যা নিশ্চিত করে যে এমনকি সবচেয়ে অত্যাধুনিক এবং আনডকুমেন্টেড (undocumented) অস্ত্রগুলোও শুধু এর ওপর এসে ধাক্কা খেয়ে মিসফায়ার (misfire) করবে বা নিষ্ক্রিয় হয়ে পড়বে।

৩.২ দৃশ্যপট ১: ফিজিক্যাল ফরেনসিক এক্সট্রাকশন (Physical Forensic Extraction)

থ্রেট ক্লাসিফিকেশন (Threat Classification): ফিজিক্যাল অ্যাক্সেস (Physical Access) / হার্ডওয়্যার এক্সপ্লয়টেশন (Hardware Exploitation)
রেড টিম অ্যাসেটস (Red Team Assets): সেলিব্রাইট ইউএফইডি প্রিমিয়াম (Cellebrite UFED Premium), গ্রে-কি (GrayKey) এক্সট্রাকশন টার্মিনাল, ইউএসবি ফরেনসিক ক্যাবলস, ফ্যারাডে এভিডেন্স ব্যাগ (Faraday evidence bag)
ব্লু টিম অ্যাসেটস (Blue Team Assets): গ্রাফিনওএস ফোর্ট্রেস ব্লুপ্রিন্ট (GrapheneOS Fortress Blueprint) (ইউএসবি-সি পোর্ট কন্ট্রোল, BFU স্টেট, Titan M2, আলফানিউমেরিক পাসওয়ার্ড)
উদ্দেশ্য বা অবজেক্টিভ (রেড টিম): একটি বাজেয়াপ্ত করা GrapheneOS ডিভাইস থেকে ব্যবহারকারীর সমস্ত ডেটা এক্সট্রাক্ট করা
উদ্দেশ্য বা অবজেক্টিভ (ব্লু টিম): এক্সট্রাকশন ডিনাই (Deny) করা। অ্যাট-রেস্ট (at rest) থাকা সমস্ত ইন্টেলিজেন্স রক্ষা করা।

অপারেশনাল কনটেক্সট (Operational Context)

টার্গেট—একজন সাংবাদিক যিনি একটি বৈরী বা হোস্টাইল জুরিসডিকশনে কাজ করছেন—তাকে একটি বর্ডার চেকপয়েন্টে আটক করা হয়েছে। তার GrapheneOS ডিভাইসটি বাজেয়াপ্ত করা হয়। বর্ডার এজেন্টের হাতে ডিভাইসটি তুলে দেওয়ার ঠিক তিন সেকেন্ড আগে সাংবাদিকটি রিবুট রুল বা Reboot Rule (সেকশন ২.১৫) এক্সিকিউট করেছিলেন। ডিভাইসটি এখন অন করা আছে, লক স্ক্রিনে আটকে আছে এবং সম্পূর্ণ কোল্ড BFU (Before First Unlock) স্টেটে রয়েছে।

যেকোনো ইনকামিং রিমোট ওয়াইপ (Remote Wipe) কমান্ড ব্লক করার জন্য ডিভাইসটিকে একটি ফ্যারাডে ব্যাগের (Faraday bag) ভেতর ঢোকানো হয় এবং একটি ডিজিটাল ফরেনসিক ল্যাবরেটরিতে নিয়ে যাওয়া হয়। একজন প্রশিক্ষিত ফরেনসিক এক্সামিনার ডিভাইসটিকে একটি Cellebrite UFED Premium টার্মিনালের সাথে কানেক্ট করেন—যা বিশ্বজুড়ে ল-এনফোর্সমেন্ট এজেন্সিগুলোর ব্যবহৃত সবচেয়ে বহুল প্রচলিত মোবাইল এক্সট্রাকশন প্ল্যাটফর্ম।

সময় বয়ে যাচ্ছে। অটো-রিবুট (Auto-Reboot) টাইমারটি ৪ ঘণ্টার জন্য সেট করা আছে।

চলুন কিল চেইনটি (Kill chain) অ্যানালাইজ করা যাক।

রেড টিম: ফেজ ১ — ইউএসবি ডেটা এক্সপ্লয়টেশন (USB Data Exploitation)

ফরেনসিক এক্সামিনার ডিভাইসটিকে ফ্যারাডে ব্যাগ থেকে বের করেন এবং ফোনের নিচে একটি স্পেশালাইজড Cellebrite USB-C ক্যাবল কানেক্ট করেন। Cellebrite UFED সফটওয়্যারটি ইনিশিয়ালাইজ হয় এবং ডিভাইসের ইউএসবি সাবসিস্টেমের (USB subsystem) সাথে একটি অটোমেটেড হ্যান্ডশেক (Automated handshake) এক্সিকিউট করার চেষ্টা করে।

একটি স্ট্যান্ডার্ড কমার্শিয়াল Android ডিভাইসে, স্ক্রিন লক করা থাকলেও ইউএসবি পোর্ট পুরোপুরি অ্যাক্টিভ থাকে। Cellebrite টার্মিনালটি সাথে সাথে ডিভাইস মডেল ডিটেক্ট করে ফেলত, ইউএসবি ডেসক্রিপ্টরগুলো এনিউমারেট (Enumerate) করত, অ্যান্ড্রয়েড ভার্সন আইডেন্টিফাই করত এবং এর লাইব্রেরিতে থাকা শত শত পরিচিত ইউএসবি ড্রাইভার এক্সপ্লয়টগুলো (USB driver exploits) সাইক্লিং বা রান করা শুরু করত। এই এক্সপ্লয়টগুলোর বেশিরভাগই ইনিশিয়াল ফুটহোল্ড (Initial foothold) পাওয়ার জন্য Android Debug Bridge (ADB) ডিমন অথবা Media Transfer Protocol (MTP) হ্যান্ডলারের দুর্বলতাকে টার্গেট করে।

এক্সামিনার হ্যান্ডশেকটি সম্পূর্ণ হওয়ার জন্য অপেক্ষা করতে থাকেন।

কিন্তু কিছুই ঘটে না।

ব্লু টিম: ডিফেন্স লেয়ার ১ — ইউএসবি-সি পোর্ট ডিনায়াল (USB-C Port Denial)

Cellebrite টার্মিনালটি রিপোর্ট করে: "**Device not detected.**"

যেহেতু GrapheneOS-এর USB-C Port Control সেটিংটি "Charging-only when locked" (সেকশন ২.১৮) হিসেবে কনফিগার করা আছে, তাই স্ক্রিন লক হওয়ার ঠিক সেই মিলিসেকেন্ডেই অপারেটিং সিস্টেম কার্নেল (Kernel) কার্যকরভাবে ইউএসবি-সি পোর্টের ডেটা লাইনগুলো বিচ্ছিন্ন করে দেয়। ডেটা চলাচলের জন্য দায়ী ফিজিক্যাল ওয়্যারগুলো (Physical wires) ড্রাইভার লেভেলে ইলেকট্রনিকভাবে ডিজেবল হয়ে যায়।

Cellebrite ক্যাবলটি ব্যাটারিতে ইলেকট্রিক্যাল কারেন্ট ডেলিভার করছে (পাওয়ার পিনগুলো অ্যাক্টিভ আছে), কিন্তু ফরেনসিক টার্মিনালটি কোনো ডেটা কানেকশন এস্টাবলিশ করতে পারছে না। ডিভাইসটি কোনো এনিউমারেট (Enumerate) করে না। এটি কোনো ইউএসবি ডেসক্রিপ্টর (USB descriptors) এক্সপোজ করে না। ক্যাবলের অন্য প্রান্তে থাকা মেশিনটির অস্তিত্বই এটি স্বীকার করে না।

Cellebrite-এর আর্সেনালে থাকা প্রতিটি ইউএসবি এক্সপ্লয়েট এক্সিকিউট করার জন্য একটি অ্যাক্টিভ ডেটা হ্যান্ডশেক প্রয়োজন। কোনো হ্যান্ডশেক ছাড়া, এক্সপ্লয়েটগুলো লোড হওয়াই শুরু করতে পারে না। রেড টিমের প্রাইমারি এক্সট্রাকশন ভেক্টরটি (Extraction vector) ডেড-অন-অ্যারাইভাল (Dead on arrival) বা শুরুতেই ব্যর্থ হয়ে যায়।

এম্পিরিক্যাল ভেরিফিকেশন (Empirical Verification): DFRWS EU 2026 কনফারেন্সে উপস্থাপিত একটি পিয়ার-রিভিউড ফরেনসিক অ্যানালাইসিসে GrapheneOS-এর বিরুদ্ধে আটটি শীর্ষস্থানীয় ফরেনসিক এক্সট্রাকশন প্ল্যাটফর্মের (Cellebrite, GrayKey এবং MSAB XRY সহ) কার্যক্ষমতা ইনভেস্টিগেট করা হয়েছিল। গবেষকরা নিশ্চিত করেছেন যে, GrapheneOS-এর জন্য Cellebrite-এর সাপোর্ট কার্যকরভাবে ২০২২ সিকিউরিটি প্যাচ লেভেলেই শেষ হয়ে গেছে। একটি ফুল্লি আপডেটেড GrapheneOS ডিভাইসের বিরুদ্ধে, এই মাল্টি-মিলিয়ন ডলারের এক্সট্রাকশন টুলগুলো আক্ষরিক অর্থেই অকেজো। MSAB XRY ফুল ফাইলসিস্টেম এক্সট্রাকশনের (Full filesystem extraction) দাবি করে, কিন্তু গবেষকরা উল্লেখ করেছেন যে এটি সম্পূর্ণ "কনসেন্ট-বেসড" (Consent-based)—অর্থাৎ ইনভেস্টিগেটরের কাছে আগে থেকেই ইউজারের পিন (PIN) থাকতে হবে, যা এই টুলটিকে হোস্টাইল এক্সট্রাকশনের জন্য অপ্রাসঙ্গিক করে তোলে।

কিল চেইন স্ট্যাটাস (Kill chain status): ফেজ ১-এ ব্রোকেন বা ব্যাহত।

রেড টিম: ফেজ ২ — লক স্ক্রিন ব্রুট ফোর্স (Lock Screen Brute Force)

ফরেনসিক এক্সামিনার Cellebrite ক্যাবলটি ডিসকানেক্ট করেন এবং স্ট্র্যাটেজি পরিবর্তন করেন। তারা একটি GrayKey টার্মিনাল কানেক্ট করেন—এটি এমন একটি ডিভাইস যা বিশেষভাবে মোবাইল লক স্ক্রিন ব্রুট-ফোর্স (Brute-force) করার জন্য তৈরি, যা অতি দ্রুত বিভিন্ন পিন কম্বিনেশন সাবমিট করতে পারে।

এক্সামিনার ধারণা করেন যে হয়তো ডিভাইসটি কোনো অল্টারনেটিভ ইনপুট মেথডের মাধ্যমে পিন সাবমিশন গ্রহণ করবে। কিছু পুরোনো Android ডিভাইসে, ফরেনসিক টুলগুলো সফলভাবে Trusted Execution Environment (TEE)-এর দুর্বলতাগুলো এক্সপ্লয়েট করে লক স্ক্রিন পুরোপুরি বাইপাস করেছে, অথবা সিকিউর এলিমেন্টের (Secure element) অথেনটিকেশন ইন্টারফেসের সাথে সরাসরি কমিউনিকেট করে প্রতি সেকেন্ডে হাজার হাজার পিন অ্যাটেম্পট (PIN attempt) সাবমিট করেছে।

GrayKey তার আক্রমণ শুরু করে।

ব্লু টিম: ডিফেন্স লেয়ার ২ — টাইটান এমং হার্ডওয়্যার রেট লিমিটিং (Titan M2 Hardware Rate Limiting)

এমনকি যদি আমরা তাত্ত্বিকভাবে ধরেও নিই যে রেড টিম কোনোভাবে ইউএসবি পোর্ট লকআউট বাইপাস করতে সক্ষম হয়েছে (হয়তো এখনো আবিষ্কৃত হয়নি এমন কোনো হার্ডওয়্যার এক্সপ্লয়টের মাধ্যমে), তবুও তাদের সামনে একটি দুর্ভেদ্য দ্বিতীয় প্রাচীর দাঁড়িয়ে থাকে: **Titan M2** সিকিউর এলিমেন্ট।

GrapheneOS-এ, লক স্ক্রিন অথেনটিকেশন মেইন অ্যাপ্লিকেশন প্রসেসর (Application Processor) দ্বারা প্রসেস হয় না। পিন বা পাসওয়ার্ডটি সরাসরি Titan M2 চিপে রাউট করা হয়—যা পিক্সেল (Pixel) মাদারবোর্ডে সোল্ডার করা একটি ফিজিক্যালি আইসোলেটেড, ট্যাম্পার-রেজিসট্যান্ট (Tamper-resistant) হার্ডওয়্যার সিকিউরিটি মডিউল।

Titan M2 চিপটি Weaver প্রটোকলের মাধ্যমে অথেনটিকেশন অ্যাটেম্পটগুলোর ওপর **অ্যাগ্রেসিভ, হার্ডওয়্যার-লেভেল রেট লিমিটিং (Hardware-level rate limiting)** প্রয়োগ করে। এই রেট লিমিটিং সফটওয়্যারে ইমপ্লিমেন্ট করা নেই (যা মডিফায়েড ফার্মওয়্যার ফ্ল্যাশ করে বাইপাস করা সম্ভব হতো); এটি চিপটির সিলিকনের ভেতরেই হার্ডকোডেড করা থাকে।

এর রেট-লিমিটিং কার্ভটি (Rate-limiting curve) নিম্নরূপ কাজ করে:

Titan M2 রেট-লিমিটিং কার্ভ (Titan M2 Rate-Limiting Curve):

- **১-৫টি ব্যর্থ প্রচেষ্টা:** কোনো ডিলে বা বিলম্ব নেই (তাৎক্ষণিক রিট্রাই)
- **৬-১০টি ব্যর্থ প্রচেষ্টা:** প্রতিটি প্রচেষ্টার মাঝে ৩০ সেকেন্ডের ডিলে
- **১১-৩০টি ব্যর্থ প্রচেষ্টা:** সূচকীয় হারে বা এক্সপোনেনশিয়ালি বর্ধিত ডিলে
- **৩০+ ব্যর্থ প্রচেষ্টা:** প্রতিটি সিঙ্গেল প্রচেষ্টার মাঝে ডিলে বেড়ে কয়েক ঘণ্টা পর্যন্ত পৌঁছায়

যদি সাংবাদিকটি একটি সাধারণ ৪-ডিজিটের নিউমেরিক পিন (১০,০০০টি সম্ভাব্য কম্বিনেশন) ব্যবহার করতেন, তবে শুধুমাত্র Titan M2-এর রেট লিমিটিং-ই ব্রুট-ফোর্স প্রক্রিয়াটিকে একটানা **কয়েক সপ্তাহ** ধরে চলতে বাধ্য করত।

যাইহোক, আমাদের সাংবাদিকটি Fortress Blueprint অনুসরণ করেছেন এবং একটি শক্তিশালী **আলফানিউমেরিক পাসওয়ার্ড** (সেকশন ১.৭) কনফিগার করেছেন। মিক্সড কেস এবং সিম্বলসহ একটি **১২**-ক্যারেক্টারের আলফানিউমেরিক পাসওয়ার্ডে প্রায় 95^12, বা মোটামুটি **৫৪০ বিলিয়ন বিলিয়ন** সম্ভাব্য কম্বিনেশনের একটি কী-স্পেস (Keyspace) থাকে।

ইনিশিয়াল লকআউট থ্রেশহোল্ড পার হওয়ার পর Titan M2-এর প্রতি কয়েক মিনিটে একটি মাত্র প্রচেষ্টার এনফোর্সড রেট অনুযায়ী, এই পাসওয়ার্ডটি ব্রুট-ফোর্স করতে মহাবিশ্বের অনুমিত তাপীয় মৃত্যু বা হিট ডেথের (Heat death of the universe) চেয়েও বেশি সময় লাগবে।

কিল চেইন স্ট্যাটাস (Kill chain status): ফেজ ২-এ ব্রোকেন বা ব্যাহত।

রেড টিম: ফেজ ৩ — র‍্যাম এক্সট্রাকশন (কোল্ড বুট) বা RAM Extraction (Cold Boot)

ফরেনসিক টিম ক্রমবর্ধমানভাবে মরিয়া হয়ে একটি অ্যাডভান্সড ফিজিক্যাল অ্যাটাকের কথা বিবেচনা করে। ডিভাইসটি যদি AFU (After First Unlock) স্টেটে থাকে, তবে এনক্রিপশন কীগুলো (Encryption keys) র‍্যামে (RAM) অবস্থান করবে। তারা একটি কোল্ড বুট (Cold Boot) অ্যাটাকের চেষ্টা করতে পারে (সেকশন ২.১৬)—মেমরিটিকে ফ্রিজ করার জন্য র‍্যামে লিকুইড নাইট্রোজেন স্প্রে করে চিপটিকে ফিজিক্যালি এক্সট্রাক্ট করা।

ব্লু টিম: ডিফেন্স লেয়ার ৩ — বিএফইউ স্টেট এবং অটো-রিবুট (BFU State and Auto-Reboot)

আক্রমণটি শুরু হওয়ার আগেই ব্যর্থ হয়ে যায়।

ডিভাইসটি সমর্পণ করার আগেই সাংবাদিকটি Reboot Rule এক্সিকিউট করেছিলেন। ফলে ডিভাইসটি **BFU (Before First Unlock)** স্টেটে রয়েছে। রিবুট সিকোয়েন্স চলাকালীন র‍্যাম সম্পূর্ণভাবে মুছে ফেলা হয়েছে (Wiped)। ফ্রিজ করার মতো মেমরিতে কোনো এনক্রিপশন কী-ই নেই। কোল্ড বুট অ্যাটাকটি শুধুমাত্র ক্রিপ্টোগ্রাফিক ভয়েড (Cryptographic void) বা শূন্যতা ছাড়া আর কিছুই এক্সট্রাক্ট করতে পারে না।

উপরন্তু, সাংবাদিক যদি ডিভাইসটি রিবুট করতে ব্যর্থও হতেন এবং এটি AFU স্টেটে থাকা অবস্থাতেই বাজেয়াপ্ত করা হতো, তাহলেও **অটো-রিবুট টাইমার (Auto-Reboot timer)** (যা ৪ ঘণ্টার জন্য কনফিগার করা) ফেইল-সেইফ (Fail-safe) হিসেবে কাজ করত। ফরেনসিক টিম ডিভাইসটিকে ল্যাবরেটরিতে নিয়ে গেছে, এভিডেন্স হিসেবে লগ ইন করেছে এবং এক্সট্রাকশনের জন্য কিউ-তে (Queue) রেখেছে। এক্সামিনার ক্যাবল কানেক্ট করার আগেই ৪ ঘণ্টা ১২ মিনিট পেরিয়ে গেছে।

GrapheneOS ইতোমধ্যে স্বয়ংক্রিয়ভাবে রিবুট হয়ে গেছে। র‍্যাম স্যানিটাইজ করা হয়েছে। ডিভাইসটি এখন BFU স্টেটে রয়েছে। ডেডম্যানস সুইচের (Deadman's Switch) মাধ্যমে সুযোগের জানালা চিরতরে বন্ধ হয়ে গেছে।

কিল চেইন স্ট্যাটাস (Kill chain status): ফেজ ৩-এ ব্রোকেন বা ব্যাহত।

রেড টিম: ফেজ ৪ — ফার্মওয়্যার ডাউনগ্রেড (Firmware Downgrade)

সর্বশেষ ব্যবস্থা হিসেবে, রেড টিম একটি পূর্ববর্তী প্যাচ করা দুর্বলতাকে এক্সপ্লয়ট করার জন্য ডিভাইসটিতে একটি পুরোনো এবং ভালনারেবল (Vulnerable) ফার্মওয়্যার ভার্সন ফ্ল্যাশ করার কথা বিবেচনা করে।

ব্লু টিম: ডিফেন্স লেয়ার ৪ — অ্যান্টি-রোলব্যাক প্রোটেকশন (Anti-Rollback Protection)

GrapheneOS, Titan M2 সিকিউর এলিমেন্টের সাথে একত্রে, কঠোর **অ্যান্টি-রোলব্যাক প্রোটেকশন (Anti-Rollback Protection)** প্রয়োগ করে। হার্ডওয়্যারটি একটি ইন্টারনাল কাউন্টার মেইনটেইন করে যা মিনিমাম অ্যাক্সেপ্টেবল বা সর্বনিম্ন গ্রহণযোগ্য ফার্মওয়্যার ভার্সন ট্র্যাক করে। বর্তমান কাউন্টারের চেয়ে পুরোনো কোনো ফার্মওয়্যার ভার্সন ফ্ল্যাশ করার যেকোনো প্রচেষ্টাকে সিলিকন স্বয়ংক্রিয়ভাবে এবং স্থায়ীভাবে প্রত্যাখ্যান করে।

রেড টিম অপারেটিং সিস্টেম ডাউনগ্রেড করতে পারে না। তারা কোনো ভালনারেবল বিল্ডও (Vulnerable build) ইনস্টল করতে পারে না। হার্ডওয়্যার নিজেই তাদের সাথে কোঅপারেট বা সহযোগিতা করতে অস্বীকৃতি জানায়।

কিল চেইন স্ট্যাটাস (Kill chain status): ফেজ ৪-এ ব্রোকেন বা ব্যাহত।

রেড টিম: ফেজ ৫ — অ্যাপ ডাউনগ্রেডিং অ্যাটাকস (App Downgrading Attacks)

অপারেটিং সিস্টেম ডাউনগ্রেড করতে ব্যর্থ হয়ে, ফরেনসিক এক্সামিনার একটি "অ্যাপ ডাউনগ্রেডিং অ্যাটাক (App Downgrading Attack)" করার চেষ্টা করেন। এটি একটি সাধারণ ফরেনসিক টেকনিক যেখানে একজন ইনভেস্টিগেটর একটি আধুনিক, সিকিউর অ্যাপ (যেমন WhatsApp বা Signal) আনইনস্টল করেন, কিন্তু এর ইউজার ডেটা ডিরেক্টরি ডিলিট করেন পরিচয় দেন না। এরপর তারা কমান্ড লাইনের মাধ্যমে সেই একই অ্যাপের একটি পুরোনো, ভালনারেবল ভার্সন জোরপূর্বক ইনস্টল করেন, যাতে খুব সহজেই এর লোকাল ডেটাবেসটি এক্সট্রাক্ট করা যায়।

ব্লু টিম: ডিফেন্স লেয়ার ৫ — স্ট্রিক্ট সিগনেচার এবং ভার্সন ভ্যালিডেশন (Strict Signature & Version Validation)

এই অ্যাটাকটি GrapheneOS-এ ধারাবাহিকভাবে ব্যর্থ হয়। যেমনটি DFRWS EU 2026 গবেষকরা ভেরিফাই করেছেন, GrapheneOS যেকোনো APK ইনস্টলেশনকে জোরালোভাবে ব্লক করে যার ভার্সন কোড বর্তমানে ইনস্টল করা অ্যাপের ভার্সনের চেয়ে কম, অথবা এমন যেকোনো APK যার ক্রিপ্টোগ্রাফিক সিগনেচার (Cryptographic signature) অরিজিনাল ডেভেলপারের সিগনেচারের সাথে হুবহু মিলে না। ওএস ডাউনগ্রেডের অনুমতি দিতে সম্পূর্ণভাবে অস্বীকৃতি জানায়, যা এই এক্সট্রাকশন ভেক্টরটিকে সম্পূর্ণরূপে নিষ্ক্রিয় করে দেয়।

কিল চেইন স্ট্যাটাস (Kill chain status): ফেজ ৫-এ ব্রোকেন বা ব্যাহত।

আফটার-অ্যাকশন রিপোর্ট (After-Action Report)

চারটি ফেজেই ফরেনসিক এক্সট্রাকশন ব্যাপকভাবে ব্যর্থ হয়েছে। রেড টিম একটি সঠিকভাবে কনফিগার করা GrapheneOS ডিভাইসের বিরুদ্ধে তাদের সবচেয়ে অত্যাধুনিক ফিজিক্যাল টুলগুলো ডিপ্লয় করেছিল, এবং প্রতিটি ভেক্টরই নিষ্ক্রিয় হয়ে গেছে—ভাগ্যের জোরে নয়, বরং সফটওয়্যার এবং হার্ডওয়্যার উভয়ের মধ্যেই তৈরি করা স্তরভিত্তিক, ম্যাথমেটিক্যাল বা গাণিতিক ডিফেন্সের কারণে।

আফটার-অ্যাকশন সামারি (After-Action Summary):

ফেজ (PHASE)	অ্যাটাক ভেক্টর (ATTACK VECTOR)	GRAPHENEOS ডিফেন্স লেয়ার (DEFENSE LAYER)	ফলাফল (RESULT)
১	ইউএসবি ডেটা এক্সপ্লয়টেশন (USB data exploitation)	ইউএসবি-সি পোর্ট কন্ট্রোল (ডেটা লাইন বিচ্ছিন্ন)	**ডিনাইড (DENIED)**
২	লক স্ক্রিন ব্রুট ফোর্স (Lock screen brute force)	Titan M2 হার্ডওয়্যার রেট লিমিটিং (Weaver)	**ডিনাইড (DENIED)**
৩	কোল্ড বুট র‍্যাম এক্সট্রাকশন (Cold Boot RAM extraction)	BFU স্টেট + অটো-রিবুট টাইমার	**ডিনাইড (DENIED)**
৪	ফার্মওয়্যার ডাউনগ্রেড (Firmware downgrade)	অ্যান্টি-রোলব্যাক প্রোটেকশন (হার্ডওয়্যার কাউন্টার)	**ডিনাইড (DENIED)**
৫	অ্যাপ ডাউনগ্রেড অ্যাটাক (App downgrade attack)	স্ট্রিক্ট সিগনেচার এবং ভার্সন ভ্যালিডেশন	**ডিনাইড (DENIED)**

৭২ ঘণ্টা আটকে রাখার পর সাংবাদিকের ডিভাইসটি ফিরিয়ে দেওয়া হয়। তিনি তার আলফানিউমেরিক পাসওয়ার্ডটি প্রবেশ করান এবং ফোর্ট্রেসটি বা দুর্গটি আনলক হয়। প্রতিটি ফাইল, প্রতিটি এনক্রিপ্ট করা মেসেজ এবং প্রতিটি কন্ট্যাক্ট ঠিক সেখানেই আছে যেখানে তিনি রেখেছিলেন। ইন্টেলিজেন্স বা ডেটা কখনোই কম্প্রোমাইজড হয়নি।

দৃশ্যপট ১: **ব্লু টিমের বিজয় (Blue Team victory)**।

৩.৩ দৃশ্যপট ২: কোল্ড বুট র‍্যাম এক্সট্রাকশন (The Cold Boot RAM Extraction)

থ্রেট ক্লাসিফিকেশন (Threat Classification): ফিজিক্যাল অ্যাক্সেস (Physical Access) / ক্রায়োজেনিক হার্ডওয়্যার এক্সপ্লয়টেশন (Cryogenic Hardware Exploitation)
রেড টিম অ্যাসেটস (Red Team Assets): কম্প্রেসড এয়ার ক্যানিস্টার (Compressed air canister), লিকুইড নাইট্রোজেন স্প্রে (liquid nitrogen spray), জেটিএজি এক্সট্রাকশন বোর্ড (JTAG extraction board), কাস্টম ডি-র‍্যাম রিডার (custom DRAM reader), পোর্টেবল ফ্রিজার ইউনিট (portable freezer unit)
ব্লু টিম অ্যাসেটস (Blue Team Assets): গ্রাফিনওএস ফোর্ট্রেস ব্লুপ্রিন্ট (GrapheneOS Fortress Blueprint) (অটো-রিবুট টাইমার, জিরো-অন-ফ্রি মেমরি পলিসি, hardened_malloc, BFU এনফোর্সমেন্ট)
উদ্দেশ্য বা অবজেক্টিভ (রেড টিম): ভোলাটাইল র‍্যাম (Volatile RAM) থেকে ফাইল-বেসড এনক্রিপশন (File-Based Encryption বা FBE) মাস্টার কী (master keys) এক্সট্রাক্ট করা
উদ্দেশ্য বা অবজেক্টিভ (ব্লু টিম): ফিজিক্যাল মেমরিতে যেন কোনো ব্যবহারযোগ্য ক্রিপ্টোগ্রাফিক ম্যাটেরিয়াল (cryptographic material) টিকে না থাকে তা নিশ্চিত করা

অপারেশনাল কনটেক্সট (Operational Context)

দৃশ্যপট ১-এ, আমরা এমন একটি ডিভাইস বিশ্লেষণ করেছি যা বাজেয়াপ্ত হওয়ার আগে সঠিকভাবে BFU স্টেটে রিবুট করা হয়েছিল। এর র‍্যাম (RAM) সম্পূর্ণ খালি ছিল এবং ফরেনসিক টিম কেবল একটি ক্রিপ্টোগ্রাফিক ভয়েডের (Cryptographic void) সম্মুখীন হয়েছিল।

কিন্তু যখন কোনো অপারেটর রিবুট রুল (Reboot Rule) এক্সিকিউট করেন না, তখন কী ঘটে?

এই দৃশ্যপটে, টার্গেট হলেন একজন কর্পোরেট এক্সিকিউটিভ, যার ডিভাইসটি একটি বিজনেস ডিনারের সময় রেস্টুরেন্টের টেবিল থেকে চুরি হয়ে যায়। ডিভাইসটি স্ক্রিন-লক করা অবস্থায় থাকলেও, এটি গত এক ঘণ্টার মধ্যেই আনলক করা হয়েছিল এবং সক্রিয়ভাবে ব্যবহৃত হয়েছিল। তাই ডিভাইসটি এখন **AFU (After First Unlock)** স্টেটে রয়েছে। ফাইল-বেসড এনক্রিপশন কীগুলো র‍্যান্ডম অ্যাক্সেস মেমরিতে (Random Access Memory বা RAM) লোড করা আছে, যা বায়োমেট্রিক বা পিনের (PIN) মাধ্যমে দ্রুত আনলক করার জন্য ফাইল সিস্টেমটিকে অ্যাক্সেসযোগ্য করে রাখে।

এখানে রেড টিম হলো একটি অত্যাধুনিক কর্পোরেট এসপিওনাজ ইউনিট (Corporate espionage unit), যাদের কাছে উন্নত ফিজিক্যাল এক্সট্রাকশন হার্ডওয়্যারের অ্যাক্সেস রয়েছে। তারা বুঝতে পারে যে এনক্রিপশন কীগুলো এখনও মেমরিতে 'হট' (hot) অবস্থায় আছে, এবং ইলেকট্রনগুলো ক্ষয় বা ডিকে (decay) হওয়ার আগেই তারা ফিজিক্যালি সিলিকন থেকে কীগুলোকে রির্প (rip) করে বের করে আনার উদ্দেশ্য নিয়ে কাজ করছে।

সময় বয়ে যাচ্ছে। অটো-রিবুট (Auto-Reboot) টাইমারটি ৪ ঘণ্টার জন্য সেট করা আছে।

রেড টিম: ডেটা রেম্যানেন্সের পদার্থবিজ্ঞান (The Physics of Data Remanence)

কোল্ড বুট (Cold Boot) অ্যাটাক বুঝতে হলে আমাদের র‍্যামের ফিজিক্স বা পদার্থবিজ্ঞানটি বিশ্লেষণ করতে হবে।

ডায়নামিক র‍্যান্ডম অ্যাক্সেস মেমরি (DRAM) ক্যাপাসিটরে অতি ক্ষুদ্র ইলেকট্রিক্যাল চার্জ হিসেবে ডেটা স্টোর করে। এই চার্জগুলো স্বভাবতই আনস্টেবল বা অস্থিতিশীল। ক্যাপাসিটরগুলো চার্জ লিক (leak) করে এবং মেমরি কন্ট্রোলারকে প্রতি সেকেন্ডে হাজার হাজার বার ডেটা ইলেকট্রনিকভাবে "রিফ্রেশ" করতে হয়। যখন কোনো ডিভাইসের পাওয়ার সংযোগ বিচ্ছিন্ন হয়, তখন রিফ্রেশ সাইকেলটি থেমে যায় এবং চার্জগুলো ডিকে (decay) বা ক্ষয় হতে শুরু করে। সাধারণ অপারেটিং কন্ডিশনে, রুমের তাপমাত্রায় (আনুমানিক 25°C), ডি-র‍্যামে থাকা ডেটা প্রায় এক থেকে তিন সেকেন্ডের মধ্যে আনরিকভারেবল নয়েজ (Unrecoverable noise)-এ পরিণত হয়।

২০০৮ সালে, প্রিন্সটন ইউনিভার্সিটির গবেষকরা একটি যুগান্তকারী গবেষণাপত্র প্রকাশ করেন, যেখানে দেখানো হয় যে এই ক্ষয়ের হার বা ডিকে রেট (Decay rate) সরাসরি সিলিকনের তাপমাত্রার সাথে সম্পর্কিত। র‍্যাম চিপগুলোকে সাব-জিরো তাপমাত্রায় ঠান্ডা করার মাধ্যমে ইলেকট্রন লিকেজের হার নাটকীয়ভাবে হ্রাস করা সম্ভব। -50°C তাপমাত্রায় ডেটা কয়েক মিনিট পর্যন্ত টিকে থাকতে পারে। আর লিকুইড নাইট্রোজেনের মাধ্যমে অর্জনযোগ্য চরম তাপমাত্রায় (-196°C), ডেটা রেম্যানেন্স (Data remanence) বা ডেটার স্থায়িত্ব এক ঘণ্টারও বেশি সময় পর্যন্ত বৃদ্ধি পেতে পারে।

রেড টিম নিচের প্রটোকলটি এক্সিকিউট করে:

১. চুরি হওয়া ডিভাইসটি সাথে সাথেই পাওয়ার ডাউন (Power down) করা হয়। ২. কয়েক সেকেন্ডের মধ্যেই, রেড টিম ডিভাইসের মাদারবোর্ডে থাকা উন্মুক্ত র‍্যাম চিপগুলোর ওপর উল্টো করে ধরা কম্প্রেসড এয়ার (compressed air) স্প্রে করে (এটি এমন একটি টেকনিক যা অতি দ্রুত সারফেসের তাপমাত্রা প্রায় -50°C এ নামিয়ে আনে)। ৩. একটি হট-এয়ার রিওয়ার্ক স্টেশন (hot-air rework station) ব্যবহার করে টার্গেট ডিভাইসের মাদারবোর্ড থেকে জমে যাওয়া র‍্যাম মডিউলটি ফিজিক্যালি ডিসোল্ডার (desoldered) করে আলাদা করা হয়। ৪. হিমায়িত বা ফ্রোজেন মডিউলটি সাথে সাথেই একটি কাস্টম

ডি-র‍্যাম রিডার বোর্ডে (custom DRAM reader board) ট্রান্সফার করা হয়—এটি এমন একটি স্পেশালাইজড সার্কিট যা মেমরি মডিউলে পাওয়ার সরবরাহ করার জন্য এবং ইলেকট্রনগুলো গলে গিয়ে ক্ষয় বা ডিকে হওয়ার আগেই সম্পূর্ণ র-বাইনারি কন্টent (raw binary contents) ডাম্প (dump) করার জন্য ডিজাইন করা হয়েছে। ৫. ডাম্প করা বাইনারিতে AES-256 এনক্রিপশন কী (Encryption key) স্ট্রাকচারের সাথে মিলে যাওয়া প্যাটার্নগুলো স্ক্যান করার জন্য রেড টিম একটি ফরেনসিক অ্যানালাইসিস টুল (Forensic analysis tool) রান করে।

একটি স্ট্যান্ডার্ড, আনমিটিগেটেড (unmitigated) অপারেটিং সিস্টেমে, এই অ্যাটাকের ঐতিহাসিকভাবে প্রমাণিত সাফল্যের হার রয়েছে। FBE কীগুলো মেমরির একটি অনুমানযোগ্য এবং ওয়েল-ডকুমেন্টেড (well-documented) অংশে বসে থাকে। ফরেনসিক টুলটি সহজেই সেগুলো খুঁজে বের করে এবং অ্যাডভার্সারি (Adversary) সেগুলোকে ব্যবহার করে অফলাইনে সম্পূর্ণ ফাইল সিস্টেমটি ডিক্রিপ্ট (Decrypt) করে ফেলে।

ব্লু টিম: ডিফেন্স লেয়ার ১ — দ্য ডেডম্যানস সুইচ (The Deadman's Switch)

ডিফেন্সের ফার্স্ট লাইনটি বা প্রথম স্তরটি হলো টেম্পোরাল বা সময়ভিত্তিক।

রেড টিমকে অবশ্যই অটো-রিবুট উইন্ডোর (Auto-Reboot window) মধ্যেই তাদের ক্রায়োজেনিক এক্সট্রাকশন (Cryogenic extraction) সম্পন্ন করতে হবে। যদি এক্সিকিউটিভের ডিভাইসটি রাত ৮:০০ টায় চুরি হয়ে থাকে এবং অটো-রিবুট টাইমারটি ৪ ঘণ্টার জন্য সেট করা থাকে, তবে GrapheneOS স্বয়ংক্রিয়ভাবে ঠিক রাত ১২:০০ টায় একটি সম্পূর্ণ সিস্টেম রিবুট ট্রিগার (Trigger) করবে।

যখন অটো-রিবুট ফায়ার করে বা চালু হয়, অপারেটিং সিস্টেমটি তখন গ্রেসফুল শাটডাউন (graceful shutdown) পারফর্ম করে না। বরং এটি একটি হার্ড রিবুট সিকোয়েন্স (hard reboot sequence) এক্সিকিউট করে, যা জোরপূর্বক র‍্যামের স্টেট ক্লিয়ার করে ফেলে। এনক্রিপশন কীগুলো অ্যাক্টিভলি জিরো (Zeroes) দিয়ে ওভাররাইট (Overwrite) করা হয় এবং তারপর হার্ডওয়্যার পাওয়ার ডাউন হয়ে যায়। ডিভাইসটি যখন রিবুট হয়ে পুনরায় লক স্ক্রিনে ফিরে আসে, তখন এটি সম্পূর্ণ ফ্রেশ একটি BFU স্টেটে থাকে, যেখানে কোনো রেসিডিউয়াল (residual) বা অবশিষ্ট কী ম্যাটেরিয়াল থাকে না।

যদি রেড টিম মধ্যরাতের আগেই ক্রায়োজেনিক এক্সট্রাকশন সম্পন্ন করতে না পারে, তবে কীগুলো চিরতরে হারিয়ে যায়। এই অ্যাটাক উইন্ডোটি সীমিত, এবং ডেডম্যানস সুইচ (Deadman's Switch) এক্ষেত্রে নির্দয়ভাবে কাজ করে।

যে অপারেটর তার অটো-রিবুট টাইমারটি কমিয়ে ১ ঘণ্টায় নিয়ে আসেন (যেমনটি সেকশন ২.১৬-এ হাই-থ্রেট এনভায়রনমেন্টের জন্য সুপারিশ করা হয়েছে), তিনি এই এক্সট্রাকশন উইন্ডোটিকে একটি প্রায় অসম্ভব এবং আঁটসাঁট অপারেশনাল টাইমলাইনে পরিণত করেন। অ্যাডভার্সারিকে ডিভাইস চুরি করতে হবে, সেটিকে ক্রায়োজেনিক ইকুইপমেন্টসহ একটি ল্যাবরেটরিতে নিয়ে

যেতে হবে, চ্যাসিসটি (chassis) খুলতে হবে এবং র‍্যাম চিপটি ডিসোল্ডার করতে হবে—এবং এই সবকিছুই করতে হবে মাত্র ৬০ মিনিটের মধ্যে।

ব্লু টিম: ডিফেন্স লেয়ার ২ — জিরো-অন-ফ্রি মেমরি স্যানিটাইজেশন (Zero-on-Free Memory Sanitization)

এমনকি অটো-রিবুট উইন্ডোর ভেতরেও, রেড টিমকে দ্বিতীয় এবং আরও অনেক বেশি ভয়াবহ একটি ডিফেন্সের সম্মুখীন হতে হয়: **জিরো-অন-ফ্রি মেমরি ক্লিয়ারিং (Zero-on-free memory clearing)**।

একটি স্ট্যান্ডার্ড Android ডিভাইসে, যখন কোনো অ্যাপ্লিকেশন মেমরির একটি ব্লক ব্যবহার শেষ করে (উদাহরণস্বরূপ, আপনি আপনার ব্যাংকিং অ্যাপটি বন্ধ করার পর), তখন অপারেটিং সিস্টেম সেই মেমরিটিকে "ফ্রি (Free)" হিসেবে মার্ক করে এবং অন্যান্য প্রসেসের জন্য সেটিকে অ্যাভেইলেবল বা সহজলভ্য করে। তবে, এটি কিন্তু বাস্তবে সেই ফ্রি বা উন্মুক্ত হওয়া মেমরি পেইজগুলোতে (Memory pages) সংরক্ষিত ডেটা ইরেজ (erase) বা মুছে ফেলে না। পুরনো ডেটাগুলো—যাতে পাসওয়ার্ডের অংশবিশেষ, সেশন টোকেন এবং ডিক্রিপশন কী থাকতে পারে—র‍্যামের মধ্যে "ঘোস্ট ডেটা (Ghost data)" হিসেবে ফিজিক্যালি তখনো পর্যন্ত অস্তিত্বশীল থাকে, যতক্ষণ না অন্য কোনো অ্যাপ্লিকেশন সেই মেমরিটি ওভাররাইট করে।

এই ঘোস্ট ডেটা কোল্ড বুট অ্যাটাকারদের জন্য একটি স্বর্ণখনি। এমনকি যদি অ্যাক্টিভ এনক্রিপশন কীগুলো রোটেট (rotate) করা হয়ে থাকে বা অ্যাপ্লিকেশনটি ক্লোজ করা হয়ে থাকে, তবুও সংবেদনশীল ক্রিপ্টোগ্রাফিক ম্যাটেরিয়ালের টুকরো বা ফ্র্যাগমেন্টগুলো (fragments) ফ্রি হওয়া মেমরি পেইজে দীর্ঘ সময়ের জন্য থেকে যেতে পারে।

GrapheneOS একটি অ্যাগ্রেসিভ **জিরো-অন-ফ্রি (Zero-on-free)** মেমরি স্যানিটাইজেশন পলিসির মাধ্যমে এই ভালনারেবিলিটি বা দুর্বলতাটিকে সম্পূর্ণ নিশ্চিহ্ন করে দেয়।

যে মুহূর্তেই কোনো প্রসেস দ্বারা একটি মেমরি পেইজ ফ্রি হয়—হোক সেটি কোনো ইউজার অ্যাপ্লিকেশন, সিস্টেম ডিমন, বা কার্নেল নিজেই—GrapheneOS তৎক্ষণাৎ এবং শর্তহীনভাবে সেই পেইজের সম্পূর্ণ কন্টেন্টকে জিরো (zeroes) দিয়ে ওভাররাইট করে ফেলে, এবং এরপরই কেবল সেটিকে অ্যাভেইলেবল মেমরি পুলে (available memory pool) ফেরত পাঠায়। এটি কার্নেল লেভেলে এনফোর্স বা প্রয়োগ করা হয়, যা কোনো ইউজার-স্পেস (User-space) অ্যাপ্লিকেশন দ্বারা বাইপাস করা সম্ভব নয়।

উপরন্তু, GrapheneOS-এর `hardened_malloc` সিস্টেমটি সমস্ত হিপ অ্যালোকেশনের (Heap allocations) জন্য **জিরো-অন-রিলিজ (Zero-on-release)** বিহেভিয়ার বা আচরণের মাধ্যমে এই নীতিটিকে আরও এক্সটেন্ড করে। যখন কোনো অ্যাপ্লিকেশন একটি বাফার ডিঅ্যালোকেট (Deallocate) করে (উদাহরণস্বরূপ, যখন Vanadium সেশন কুকি আছে এমন একটি জাভাস্ক্রিপ্ট অবজেক্ট প্রসেস করা শেষ করে), তখন `hardened_malloc` শুধু বাফারটিকে অ্যাভেইলেবল বা উন্মুক্ত হিসেবে মার্ক করেই কাজ শেষ করে না। এটি তাৎক্ষণিকভাবে সেই বাফারের প্রতিটি বাইটকে জিরো দিয়ে ওভাররাইট করে, যাতে নিশ্চিত করা যায় যে কোনো রেসিডিউয়াল বা অবশিষ্ট ডেটাই আর টিকে নেই।

কোল্ড বুট অ্যাটাকের ওপর এর ব্যবহারিক প্রভাব রেড টিমের জন্য ধ্বংসাত্মক।

এমনকি যদি তারা অটো-রিবুট উইন্ডোর মধ্যেই সফলভাবে র‍্যাম ফ্রিজ করে এবং বাইনারি কন্টেন্ট ডাম্প করতে পারে, তবুও এক্সট্রাক্ট করা মেমরি ইমেজে দেখা যাবে সেটি স্যানিটাইজ করা এবং জিরো-আউট (zeroed-out) পেইজে ভর্তি। এনক্রিপশন কীগুলো কোনো বিস্তীর্ণ ঘোস্ট ডেটার মাঠে আবিষ্কৃত হওয়ার অপেক্ষায় অলসভাবে বসে থাকে না। এরা শুধুমাত্র অত্যন্ত সংকীর্ণ এবং অ্যাক্টিভলি ব্যবহৃত মেমরি রিজিওন বা অঞ্চলে অস্তিত্বশীল থাকে, যার চারপাশ ঘিরে থাকে সম্পূর্ণ গাণিতিকভাবে ক্লিন জিরো (Zeroes)-এর একটি অনুর্বর প্রান্তর বা ওয়েস্টল্যান্ড (wasteland)।

ফলস্বরূপ, ফরেনসিক সার্চ স্পেসটি (Forensic search space) মারাত্মকভাবে কমে যায়, এবং অক্ষত, ব্যবহারযোগ্য কী ম্যাটেরিয়াল রিকভার করার সম্ভাবনাও একদম খাড়াভাবে নিচে নেমে আসে।

আফটার-অ্যাকশন রিপোর্ট (After-Action Report)

কোল্ড বুট অ্যাটাক হলো ফরেনসিক আর্সেনালের সবচেয়ে ফিজিক্যালি অত্যাধুনিক বা ফিজিক্যালি সফিস্টিকেটেড (Physically sophisticated) এক্সপ্লয়টগুলোর মধ্যে একটি। এর জন্য স্পেশালাইজড ইকুইপমেন্ট, নিখুঁত টাইমিং এবং সেমিকন্ডাক্টর ফিজিক্স (Semiconductor physics) সম্পর্কে গভীর জ্ঞানের প্রয়োজন হয়। একটি স্ট্যান্ডার্ড কমার্শিয়াল অপারেটিং সিস্টেমের বিরুদ্ধে নিয়ন্ত্রিত পরিস্থিতিতে এটি যে সফল হয়, তা এর আগেই প্রমাণিত হয়েছে।

তবে একটি সঠিকভাবে কনফিগার করা GrapheneOS ডিভাইসের বিরুদ্ধে, এই অ্যাটাক একটি লেয়ার্ড ডিফেন্স-ইন-ডেপ্থের (Layered defense-in-depth) সম্মুখীন হয়:

ডিফেন্স-ইন-ডেপ্থ সামারি (Defense-in-Depth Summary):

- **অটো-রিবুট টাইমার (Auto-Reboot Timer):** একটি কনফিগারেবল বা সেট করা উইন্ডোর পর জোরপূর্বক BFU স্টেটে নিয়ে যায়। এক্সট্রাকশন টাইমলাইন অতিক্রম করলে কীগুলো (Keys) সম্পূর্ণরূপে ধ্বংস করে ফেলে।
- **জিরো-অন-ফ্রি কার্নেল (Zero-on-Free Kernel):** ফ্রি হওয়া মেমরি পেইজগুলোকে সাথে সাথেই জিরো দিয়ে ওভাররাইট করে। ঘোস্ট ডেটা এবং কী ফ্র্যাগমেন্টগুলো (key fragments) পুরোপুরি এলিমিনেট করে।
- **জিরো-অন-রিলিজ হার্ডেনড ম্যালক (Zero-on-Release hardened_malloc):** ডিঅ্যালোকেশনের সময় হিপ অ্যালোকেশনগুলোকে (heap allocations) জিরো দিয়ে ওভাররাইট করে। অ্যাপ্লিকেশন-লেভেল ক্রিপ্টোগ্রাফিক রেসিডিউ (cryptographic residue) বা অবশিষ্টাংশ স্যানিটাইজ করে।
- **বিএফইউ এনফোর্সমেন্ট (BFU Enforcement):** সঠিক পিন (PIN) প্রবেশ না করানো পর্যন্ত কীগুলো কখনোই লোড হয় না। ফার্স্ট আনলকের (first unlock) আগে র‍্যামে কোনো কী ম্যাটেরিয়ালের অস্তিত্বই থাকে না।

রেস্টুরেন্ট থেকে চুরি হওয়া এক্সিকিউটিভের ডিভাইসটি চার ঘণ্টা পর অ্যাডভার্সারি বা শত্রুর গাড়ির পেছনে থাকা অবস্থাতেই স্বয়ংক্রিয়ভাবে রিবুট হয়ে যায়। র‍্যাম স্যানিটাইজ হয়ে গিয়েছিল। তাই ফরেনসিক ডাম্প (Forensic dump) জিরো বা শূন্য ছাড়া আর কিছুই ফেরত দিতে পারেনি।

দৃশ্যপট ২: **ব্লু টিমের বিজয় (Blue Team victory)**।

৩.৪ দৃশ্যপট ৩: জিরো-ক্লিক ওয়ারফেয়ার (Zero-Click Warfare) (Pegasus)

থ্রেট ক্লাসিফিকেশন (Threat Classification): রিমোট (Remote) / জিরো-ইন্টারঅ্যাকশন এক্সপ্লয়ট চেইন (Zero-Interaction Exploit Chain)
রেড টিম অ্যাসেটস (Red Team Assets): এনএসও গ্রুপের পেগাসাস-ক্লাস স্পাইওয়্যার প্ল্যাটফর্ম (NSO Group Pegasus-class spyware platform), ইমেজ রেন্ডারিং লাইব্রেরিকে টার্গেট করা জিরো-ক্লিক এক্সপ্লয়ট (zero-click exploit targeting image rendering library), কার্নেল প্রিভিলেজ এসক্যালেশন পেলোড (kernel privilege escalation payload), পারসিস্টেন্ট ইমপ্ল্যান্ট মডিউল (persistent implant module)
ব্লু টিম অ্যাসেটস (Blue Team Assets): গ্রাফিনওএস ফোর্ট্রেস ব্লুপ্রিন্ট (GrapheneOS Fortress Blueprint) (hardened_malloc, Control Flow Integrity, hardened SELinux, seccomp-bpf, disabled JIT)
উদ্দেশ্য বা অবজেক্টিভ (রেড টিম): ডিভাইসে সাইলেন্ট এবং পারসিস্টেন্ট রুট-লেভেল অ্যাক্সেস (root-level access) অর্জন করা এবং সমস্ত কমিউনিকেশন, লোকেশন ডেটা এবং মাইক্রোফোন অডিও এক্সফিলট্রেট (exfiltrate) বা চুরি করা
উদ্দেশ্য বা অবজেক্টিভ (ব্লু টিম): প্রিভিলেজ এসক্যালেশনের (Privilege escalation) আগেই এক্সপ্লয়ট কিল চেইনটি (Exploit kill chain) ভেঙে দেওয়া

অপারেশনাল কনটেক্সট (Operational Context)

এটি সেই দৃশ্যপট যা বিশ্বের ইন্টেলিজেন্স প্রফেশনাল, সাংবাদিক এবং রাষ্ট্রপ্রধানদের রাতের ঘুম কেড়ে নেয়।

২০২১ সালে, পেগাসাস প্রজেক্ট (Pegasus Project) নামে পরিচিত গ্লোবাল জার্নালিজম কনসোর্টিয়াম প্রকাশ করে যে NSO গ্রুপ—যা একটি প্রাইভেট ইসরায়েলি সাইবার-ইন্টেলিজেন্স কন্ট্রাক্টর—এমন একটি মিলিটারি-গ্রেড স্পাইওয়্যার (military-grade spyware) তৈরি এবং বিক্রি করেছে, যা টার্গেটের সম্পূর্ণ জিরো ইন্টারঅ্যাকশন (zero interaction) বা কোনোপ্রকার সংশ্লিষ্টতা ছাড়াই স্বয়ংক্রিয়ভাবে iPhone এবং Android ডিভাইসগুলোকে সাইলেন্টলি কম্প্রোমাইজ করতে সক্ষম। ক্লিক করার জন্য কোনো ম্যালিসিয়াস লিংক নেই। খোলার জন্য কোনো ফিশিং ইমেইল নেই। টার্গেটের ফোনটি কেবল একটি বিশেষভাবে তৈরি বা ক্র্যাফটেড ডেটা প্যাকেট (crafted data packet) রিসিভ করে, এবং এরপরই ডিভাইসটি সাইলেন্টলি, অদৃশ্যভাবে এবং পুরোপুরিভাবে অ্যাটাকারের নিয়ন্ত্রণে চলে যায়।

পেগাসাস-ক্লাস (Pegasus-class) এক্সপ্লয়টগুলো অফেন্সিভ সাইবার সক্ষমতার (Offensive cyber capability) চূড়ান্ত শিখরকে তুলে ধরে। এগুলো মূলত রাজনৈতিক ভিন্নমতাবলম্বী, অনুসন্ধানী সাংবাদিক, মানবাধিকার আইনজীবী এবং প্রতিদ্বন্দ্বী রাষ্ট্রপ্রধানদের বিরুদ্ধে ডিপ্লয় (Deploy) করা হয়। প্রতিটি ডিপ্লয়মেন্টের পেছনে মিলিয়ন মিলিয়ন ডলার ব্যয় হয় এবং এগুলো ভালনারেবিলিটি রিসার্চের (Vulnerability research) ক্ষেত্রে একদম কাটিং-এজ (cutting-edge) প্রযুক্তির প্রতিনিধিত্ব করে।

চলুন একটি GrapheneOS ডিভাইসের ওপর পেগাসাস-ক্লাস অ্যাসল্ট (Pegasus-class assault) সিমুলেট করা যাক এবং ফরেনসিকভাবে এর কিল চেইনটি ব্যবচ্ছেদ বা ডিসেক্ট (Dissect) করি।

রেড টিম: কিল চেইন আর্কিটেকচার (The Kill Chain Architecture)

জিরো-ক্লিক এক্সপ্লয়ট কোনো একটিমাত্র ভালনারেবিলিটি বা দুর্বলতা নয়। বরং এটি একাধিক, সিকোয়েনশিয়াল বা ধারাবাহিক এক্সপ্লয়টের একটি সুনির্দিষ্টভাবে অরকেস্ট্রেটেড চেইন (Orchestrated chain), যার প্রতিটি ধাপ তার পূর্ববর্তী ধাপের সাফল্যের ওপর ভিত্তি করে তৈরি হয়। চেইনের কোনো একটি সিঙ্গেল লিংকও যদি ব্যর্থ হয়, তবে পুরো অপারেশনটিই ধসে পড়ে।

পেগাসাস-ক্লাস কিল চেইনটি সাধারণত চারটি স্টেজের সমন্বয়ে গঠিত:

স্টেজ ১ — ইনিশিয়াল অ্যাক্সেস (জিরো-ক্লিক ট্রিগার) বা Initial Access (The Zero-Click Trigger): রেড টিম একটি ইমেজ রেন্ডারিং লাইব্রেরিতে (যেমন `libheif`, `libjxl` অথবা পুরোনো `libwebp` কোডেক) একটি ভালনারেবিলিটি খুঁজে বের করে। এরপর তারা একটি ম্যালিসিয়াস ইমেজ ফাইল ক্র্যাফট করে—যা হয়তো একটি `.webp` বা `.jbig2` কমপ্রেসড ইমেজ—যার ভেতরে একটি যত্নসহকারে তৈরি করা ডেটা পেলোড (Data payload) থাকে। যখন ডিভাইসের মিডিয়া পার্সার (media parser) এটিকে ডিকোড করার চেষ্টা করে, তখনই এই পেলোডটি একটি **বাফার ওভারফ্লো (buffer overflow)** ট্রিগার করার জন্য ডিজাইন করা থাকে।

এই ওয়েপনাইজড ইমেজটি (Weaponized image) একটি ইনভিজিবল বা অদৃশ্য, সাইলেন্ট চ্যানেলের মাধ্যমে ডেলিভার করা হয়। স্ট্যান্ডার্ড Android-এর ক্ষেত্রে, এটি হতে পারে একটি MMS মেসেজ যা নোটিফিকেশন শেড-এ স্বয়ংক্রিয়ভাবে ডাউনলোড এবং রেন্ডার হয়, অথবা একটি পুশ নোটিফিকেশন (Push notification) যার মধ্যে একটি রিচ মিডিয়া প্রিভিউ (rich media preview) থাকে। টার্গেট এটি কখনোই দেখতে পান না। ব্যাকগ্রাউন্ডেই মিডিয়া কোডেক (media codec) এই ম্যালিসিয়াস ফাইলটিকে পার্সিং (Parsing) করা শুরু করে দেয়।

স্টেজ ২ — কোড এক্সিকিউশন (শেলকোড ইনজেকশন) বা Code Execution (Shellcode Injection): যখন বাফার ওভারফ্লোটি ট্রিগার হয়, তখন রেড টিমের পেলোডটি ইমেজ রেন্ডারিং প্রসেসের একটি ক্রিটিক্যাল মেমরি পয়েন্টারকে (Critical memory pointer) ওভাররাইট করে ফেলে। একটি স্ট্যান্ডার্ড, আনমিটিগেটেড (unmitigated) সিস্টেমে, অ্যাটাকার প্রোগ্রামটির এক্সিকিউশন ফ্লো (Execution flow) হাইজ্যাক করে নেয়—অর্থাৎ তারা প্রোগ্রামের লিজিটিমেট বা বৈধ কোড থেকে একে ডাইভার্ট করে অ্যাটাকারের ইনজেক্ট করা শেলকোডের (Shellcode) দিকে রিডিরেক্ট করে (এই শেলকোড হলো ম্যালিসিয়াস ইমেজ ডেটার ভেতরে এমবেড করা একটি অতি ক্ষুদ্র প্রোগ্রাম)।

স্টেজ ৩ — প্রিভিলেজ এসক্যালেশন (স্যান্ডবক্স এস্কেপ) বা Privilege Escalation (Sandbox Escape): এই শেলকোডটি স্যান্ডবক্সড মিডিয়া রেন্ডারিং প্রসেসের (sandboxed media rendering process) ভেতরে এক্সিকিউট হয়, যার পারমিশন অত্যন্ত সীমিত থাকে। রেড টিমকে এখন অ্যাপ্লিকেশন স্যান্ডবক্স থেকে বেরিয়ে আসার জন্য এবং রুট-লেভেল প্রিভিলেজ (root-level privileges) অর্জন করার জন্য একটি দ্বিতীয় এক্সপ্লয়েট—অর্থাৎ একটি কার্নেল ভালনারেবিলিটি (kernel vulnerability)—ডিপ্লয় করতে হবে।

স্টেজ ৪ — পারসিস্টেন্স (ইমপ্ল্যান্ট) বা Persistence (The Implant): রুট অ্যাক্সেস (Root access) পাওয়ার পর, রেড টিম পেগাসাস ইমপ্ল্যান্ট (Pegasus implant) ইনস্টল করে। স্পাইওয়্যারটি মাইক্রোফোন, ক্যামেরা, জিপিএস (GPS) এবং এনক্রিপ্টেড মেসেজিং ডেটাবেসে হুক (hook) করে বা প্রবেশ করে। এরপর এটি অত্যন্ত নিভৃতে সমস্ত ডেটাকে একটি রিমোট কমান্ড-অ্যান্ড-কন্ট্রোল (Command-and-control) সার্ভারে এক্সফিলট্রেট করা শুরু করে। টার্গেট বিন্দুবিসর্গও জানতে পারেন না।

একটি স্ট্যান্ডার্ড কমার্শিয়াল ডিভাইসে, এই পুরো চেইনটি মাত্র তিন সেকেন্ডেরও কম সময়ের মধ্যে একেবারে ফ্লোলেসলি (flawlessly) বা ত্রুটিহীনভাবে এক্সিকিউট হয়ে যায়।

ব্লু টিম: চেইন ব্রেকিং (Breaking the Chain)

GrapheneOS-এ, এই কিল চেইনটি আক্ষরিক অর্থেই টুকরো টুকরো হয়ে যায়। চলুন প্রতিটি স্টেজ ধাপে ধাপে বিশ্লেষণ করা যাক।

ব্লু টিম: স্টেজ ১-এ ডিফেন্স — hardened_malloc

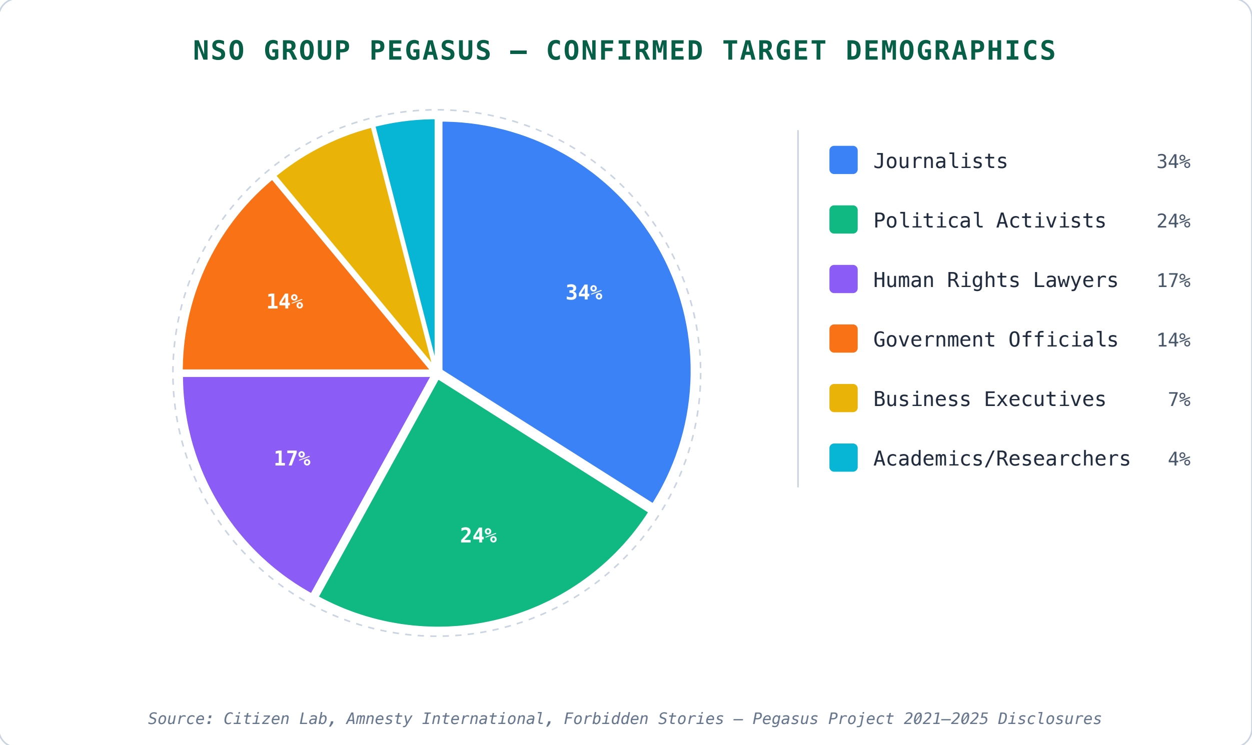
ওয়েপনাইজড ইমেজটি ডিভাইসে আসে। মিডিয়া কোডেক এটিকে পার্সিং শুরু করে। ঠিক রেড টিমের ডিজাইন অনুযায়ী ম্যালিসিয়াস ডেটাটি বাফার ওভারফ্লো (Buffer overflow) ট্রিগার করে।

তবে, বাফার ওভারফ্লোটি ক্লিনলি ল্যান্ড (cleanly land) করতে পারে না।

GrapheneOS স্ট্যান্ডার্ড Android মেমরি অ্যালোকেটর (Scudo)-কে `hardened_malloc` দিয়ে রিপ্লেস বা প্রতিস্থাপন করে। এই কাস্টম অ্যালোকেটর (Custom allocator) মেমরি কীভাবে অর্গানাইজড থাকে সেটির ফান্ডামেন্টাল আর্কিটেকচার বা মৌলিক কাঠামোটিকেই নতুন করে সাজায়। এটি মেমরি অ্যাড্রেসগুলোর স্ট্রং র‍্যান্ডমাইজেশন (strong randomization), অ্যালোকেশনগুলোর মাঝে স্ট্রিক্ট গার্ড পেইজ (strict guard pages) এবং কঠোর সাইজ-ক্লাস আইসোলেশন (size-class isolation) ইন্ট্রোডিউস বা প্রবর্তন করে।

যখন রেড টিমের পেলোডটি তার জন্য নির্ধারিত বাফারকে ওভারফ্লো করার এবং পাশের মেমরি পয়েন্টারটিকে ওভাররাইট করার চেষ্টা করে, তখন `hardened_malloc` সাথে সাথেই এই ইলিগ্যাল বাউন্ডারি ভায়োলেশন (illegal boundary violation) ডিটেক্ট করে ফেলে। বাফারের ঠিক পাশেই থাকা মেমরি পেইজটি আসলে একটি গার্ড রিজিওন (Guard region)—যা মেমরির একটি ইচ্ছাকৃত আনম্যাপড জোন (Unmapped zone)। ওভারফ্লোটি সরাসরি এই গার্ড পেইজে আছড়ে পড়ে (Crashes), যা তাৎক্ষণিক এবং আনকন্ডিশনাল বা শর্তহীন প্রসেস টার্মিনেশন (Process termination) ট্রিগার করে।

যেসকল ডিভাইসে ARM মেমরি ট্যাগিং এক্সটেনশন বা ARM Memory Tagging Extension (MTE) আছে, সেখানে এই ডিফেন্সটি আরও বেশি সার্জিক্যাল (surgical) বা নিখুঁত। প্রতিটি মেমরি অ্যালোকেশনকে একটি ছোট, র‍্যান্ডমাইজড ক্রিপ্টোগ্রাফিক লেবেল (Cryptographic label) দিয়ে ট্যাগ করা থাকে। যখন ওভারফ্লোটি তার বাউন্ডারির বাইরে গিয়ে ডেটা রাইট (Write) করে, তখন ওভারফ্লো হওয়া রিজিওনের ট্যাগটি সিপিইউ (CPU)-র এক্সপেক্ট করা ট্যাগের সাথে মেলে না। হার্ডওয়্যারটি নিজেই একটি ফল্ট (Fault) রেইজ করে, এবং প্রসেসটি মাত্র কয়েক ন্যানোসেকেন্ডের মধ্যে কিল (Killed) হয়ে যায়।



মিডিয়া রেন্ডারিং প্রসেসটি সাইলেন্টলি ক্র্যাশ (Crash) করে। ব্যবহারকারী বড়জোর ক্ষণিকের জন্য দেখতে পারেন যে একটি নোটিফিকেশন রেন্ডার হতে ব্যর্থ হয়েছে। এক্সপ্লয়ট পেলোডটি কখনোই এক্সিকিউট হয় না।

কিল চেইন স্ট্যাটাস (Kill chain status): স্টেজ ১-এ ব্রোকেন বা ব্যাহত।

ব্লু টিম: স্টেজ ২-এ ডিফেন্স — কন্ট্রোল ফ্লো ইন্টিগ্রিটি (Control Flow Integrity বা CFI)

চলুন, কঠোর বিশ্লেষণের স্বার্থে আমরা ধরে নিই যে রেড টিমের কাছে এমন একটি অ্যাডভান্সড এক্সপ্লয়ট রয়েছে যা `hardened_malloc`-কেও বাইপাস করতে সক্ষম—হয়তো মেমরি লেআউটটিকে নিখুঁতভাবে সাজানোর জন্য তারা একটি অত্যাধুনিক হিপ-গ্রুমিং (Heap-grooming) টেকনিক ব্যবহার করেছে।

বাফার ওভারফ্লো সফল হয়। মেমরি পয়েন্টার ওভাররাইট হয়ে যায়। রেড টিম এখন প্রোগ্রামটির এক্সিকিউশন ফ্লো-কে (Execution flow) তাদের ইনজেক্ট করা শেলকোডের (Shellcode) দিকে রিডিরেক্ট করার চেষ্টা করে।

GrapheneOS এক্ষেত্রে **কন্ট্রোল ফ্লো ইন্টিগ্রিটি (Control Flow Integrity বা CFI)** ডিপ্লয় করে—যা একটি কম্পাইলার-লেভেল এনফোর্সমেন্ট মেকানিজম (compiler-level enforcement mechanism) এবং এটি চলমান প্রতিটি প্রসেসের এক্সিকিউশন পাথকে (Execution path) গাণিতিকভাবে ভ্যালিডেট (Validate) করে।

যখন কোডটি CFI এর মাধ্যমে কম্পাইল করা হয়, তখন কম্পাইলার সমস্ত লিজিটিমেট বা বৈধ ফাংশন কল টার্গেটগুলোর একটি ক্রিপ্টোগ্রাফিক ম্যাপ (Cryptographic map) জেনারেট করে। রানটাইমের (Runtime) সময়, সিপিইউ (CPU) নতুন কোনো ফাংশনে জাম্প করার আগে, CFI মেকানিজম ভেরিফাই করে যে টার্গেট অ্যাড্রেসটি লিজিটিমেট ডেস্টিনেশনের প্রি-কম্পিউটেড (pre-computed) লিস্টে আছে কি না।

রেড টিমের শেলকোডটি একটি হিপ বাফারে (Heap buffer) বসে থাকে। এটি কোনো লিজিটিমেট ফাংশন নয়। এর অ্যাড্রেসটি CFI হোয়াইটলিস্টে (Whitelist) থাকে না। যখন হাইজ্যাক করা পয়েন্টারটি এক্সিকিউশনকে শেলকোডের দিকে রিডিরেক্ট করার চেষ্টা করে, তখন সাথে সাথে CFI চেক ফায়ার করে। অ্যাড্রেসটি ইনভ্যালিড (Invalid) হিসেবে সাব্যস্ত হয় এবং প্রসেসটি তাৎক্ষণিক ও আনকন্ডিশনালি টার্মিনেট বা বন্ধ হয়ে যায়।

ফলে শেলকোডটি এর একটি সিঙ্গেল ইনস্ট্রাকশনও (Single instruction) এক্সিকিউট করতে পারে না।

কিল চেইন স্ট্যাটাস (Kill chain status): স্টেজ ২-এ ব্রোকেন বা ব্যাহত।

ব্লু টিম: স্টেজ ৩-এ ডিফেন্স — হার্ডেনড এসই-লিনাক্স এবং সেককং-বিপিএফ (Hardened SELinux and seccomp-bpf)

পূর্ণাঙ্গ রিগর (Rigor) বা দৃঢ় বিশ্লেষণের স্বার্থে, চলুন তাত্ত্বিকভাবে ধরে নিই যে রেড টিমের কাছে এমন একটি এক্সপ্লয়েট রয়েছে যা `hardened_malloc` এবং CFI—উভয়টিকেই বাইপাস করতে সক্ষম—যা এমন এক অসামান্য ব্যয়বহুল সক্ষমতা, যা তৈরি করতে কয়েক মিলিয়ন ডলার খরচ হওয়ার কথা।

শেলকোডটি মিডিয়া রেন্ডারিং প্রসেসের ভেতরেই এক্সিকিউট হয়। রেড টিম এখন অ্যাপ্লিকেশন স্যান্ডবক্স (Application sandbox) থেকে পালানোর জন্য একটি কার্নেল ভালনারেবিলিটি এক্সপ্লয়েট করে প্রিভিলেজ এসক্যালেশন করার চেষ্টা করে।

GrapheneOS একটি **হার্ডেনড SELinux পলিসি (hardened SELinux policy)** ডিপ্লয় করে, যা স্ট্যান্ডার্ড Android-এর তুলনায় উল্লেখযোগ্যভাবে বেশি রেস্ট্রিক্টিভ (restrictive) বা সীমাবদ্ধতাপূর্ণ। মিডিয়া রেন্ডারিং প্রসেসটি একটি অত্যন্ত সংকীর্ণ SELinux ডোমেইনের (SELinux domain) মধ্যে সীমাবদ্ধ থাকে। প্রিভিলেজ এসক্যালেশন এক্সপ্লয়েটগুলোর

জন্য সাধারণত যেসব কার্নেল ইন্টারফেসের প্রয়োজন হয়—যেমন `ptrace`, ডিরেক্ট ডিভাইস নোড অ্যাক্সেস (direct device node access), অথবা কার্নেল মডিউল লোড করা—সেগুলোতে অ্যাক্সেস করা এর জন্য সম্পূর্ণ নিষিদ্ধ থাকে।

উপরন্তু, GrapheneOS একটি অ্যাগ্রেসিভ **সেককম্প-বিপিএফ (seccomp-bpf)** বা Secure Computing with Berkeley Packet Filter সিস্টেম কল ফিল্টারিং এনফোর্স করে। ডিভাইসের প্রতিটি প্রসেস শুধুমাত্র পারমিটেড বা অনুমোদিত সিস্টেম কলের (System calls) একটি মিনিমাল হোয়াইটলিস্টের (minimal whitelist) মধ্যেই সীমাবদ্ধ থাকে। রেড টিমের এসক্যালেশন এক্সপ্লয়ট যে কার্নেল ভালনারেবিলিটিকে টার্গেট করে, সেটির জন্য একটি স্পেসিফিক এবং অবস্কিউর (obscure) সিস্টেম কল ইনভোক (invoke) করার প্রয়োজন হয়। GrapheneOS-এর seccomp-bpf পলিসির অধীনে, সেই সিস্টেম কলটি হোয়াইটলিস্টে থাকে না। ফলে কার্নেল এটি এক্সিকিউট করতে অস্বীকার করে। এসক্যালেশন এক্সপ্লয়টটি একটি এরর কোড (Error code) রিটার্ন করে এবং চূড়ান্তভাবে ব্যর্থ হয়।

এমনকি রেড টিম যদি স্যান্ডবক্সের ভেতরে কোড এক্সিকিউশন অর্জনও করে, তবুও তারা সেখানে আটকা পড়ে যায়। স্যান্ডবক্সের দেয়ালগুলো হার্ডেনড SELinux ম্যান্ডেটরি অ্যাক্সেস কন্ট্রোল (mandatory access controls) দিয়ে রিইনফোর্সড (reinforced) বা মজবুত করা থাকে, এবং এর মেঝেটি seccomp-bpf সিস্টেম কল ফিল্টার দিয়ে সিল (sealed) করা থাকে। ফলে রুট বা root-এ যাওয়ার কোনো পথই আর বাকি থাকে না।

কিল চেইন স্ট্যাটাস (Kill chain status): স্টেজ ৩-এ ব্রোকেন বা ব্যাহত।

আফটার-অ্যাকশন রিপোর্ট (After-Action Report)

পেগাসাস-ক্লাস জিরো-ক্লিক এক্সপ্লয়ট হলো মডার্ন সাইবার আর্সেনালের এককভাবে সবচেয়ে বিপজ্জনক একটি ওয়েপন বা অস্ত্র। স্ট্যান্ডার্ড কমার্শিয়াল ডিভাইসগুলোর বিরুদ্ধে, এর ফোর-স্টেজ (four-stage) বা চার-ধাপের কিল চেইনটি মাত্র কয়েক সেকেন্ডের মধ্যে এক্সিকিউট হয়, যা অ্যাডভার্সারিকে ডিভাইসের সম্পূর্ণ, অদৃশ্য এবং পারসিস্টেন্ট কন্ট্রোল (persistent control) বা নিয়ন্ত্রণ প্রদান করে।

কিন্তু GrapheneOS-এর বিরুদ্ধে, এই চেইনটি একাধিক স্বাধীন লেয়ার বা স্তরে ভেঙে চুরমার হয়ে যায়:

কিল চেইন আফটার-অ্যাকশন সামারি (Kill Chain After-Action Summary):

- **স্টেজ ১ — মিডিয়া কোডেকে বাফার ওভারফ্লো (Buffer overflow in media codec):** hardened_malloc গার্ড পেইজ + MTE দ্বারা পরাজিত। **প্রসেস টার্মিনেটেড (Process terminated)।**

- **স্টেজ ২ — এক্সিকিউশন ফ্লো হাইজ্যাক করে শেলকোড রান (Execution flow hijack to shellcode):** কন্ট্রোল ফ্লো ইন্টিগ্রিটি (CFI) দ্বারা পরাজিত। **জাম্প রিজেক্টেড (Jump rejected), প্রসেস কিলড (process killed)।**
- **স্টেজ ৩ — স্যান্ডবক্স এস্কেপের জন্য কার্নেল এক্সপ্লয়ট (Kernel exploit for sandbox escape):** হার্ডেনড SELinux + seccomp-bpf দ্বারা পরাজিত। **সিস্টেম কল ডিনাইড (System call denied)।**
- **স্টেজ ৪ — পারসিস্টেন্ট ইমপ্ল্যান্ট ইনস্টলেশন (Persistent implant installation):** *নেভার রিচড (Never reached) বা কখনোই পৌঁছাতে পারেনি।* **প্রযোজ্য নয় (N/A)।**

রেড টিম একটি মাল্টি-মিলিয়ন-ডলার, নেশন-স্টেট-গ্রেড (nation-state-grade) এক্সপ্লয়ট চেইন ব্যয় করেছিল। এর ফলাফল ছিল কেবল একটি ব্যাকগ্রাউন্ড মিডিয়া রেন্ডারিং প্রসেসের সংক্ষিপ্ত ও ইনভিজিবল বা অদৃশ্য ক্র্যাশ (Crash)। কোনো ডেটা এক্সফিলট্রেট করা সম্ভব হয়নি। কোনো মাইক্রোফোন অ্যাক্টিভেট হয়নি। কোনো ইমপ্ল্যান্ট ইনস্টল হয়নি। অপারেটর সম্পূর্ণ অজানাই রয়ে গেলেন যে এমন কোনো অ্যাটেম্পট (Attempt) বা চেষ্টাও আদৌ করা হয়েছিল।

GrapheneOS-এর এটি জানার কোনো প্রয়োজনই ছিল না যে এই নির্দিষ্ট ভালনারেবিলিটি বা দুর্বলতাটির কোনো অস্তিত্ব আছে কি না। এর জন্য কোনো প্যাচ (Patch)-এরও প্রয়োজন পড়েনি। এই লেয়ার্ড এক্সপ্লয়ট মিটিগেশনগুলো (Layered exploit mitigations) আর্কিটেকচারাল লেভেলে (architectural level) আক্রমণটিকে নিউট্রালাইজ বা নিষ্ক্রিয় করে দিয়েছে—যা কোনো একটি নির্দিষ্ট ইন্ডিভিজুয়াল বাগকে (individual bug) নয়, বরং সম্পূর্ণ এক্সপ্লয়ট ক্লাসটিকেই (Exploit class) কিল করে।

দৃশ্যপট ৩: **ব্লু টিমের বিজয় (Blue Team victory)।**

৩.৫ ডিপ ডাইভ: এআরএম মেমরি ট্যাগিং এক্সটেনশন (ARM Memory Tagging Extension বা MTE)

পূর্ববর্তী দৃশ্যপটে, আমরা দেখেছি কীভাবে GrapheneOS-এর `hardened_malloc` এবং কন্ট্রোল ফ্লো ইন্টিগ্রিটি (Control Flow Integrity) একটি পেগাসাস-ক্লাস জিরো-ক্লিক এক্সপ্লয়টকে (Pegasus-class zero-click exploit) নিউট্রালাইজ বা নিষ্ক্রিয় করে। আমরা সেখানে সংক্ষেপে উল্লেখ করেছিলাম যে আধুনিক Pixel হার্ডওয়্যারে (Pixel 8 এবং এর পরের মডেলগুলোতে), এই ডিফেন্সটিকে সম্পূর্ণ ভিন্ন একটি মাত্রায় উন্নীত করা হয়েছে—যা শুধুমাত্র সফটওয়্যারের মাধ্যমে নয়, বরং সিপিইউ (CPU)-র ফিজিক্যাল ট্রানজিস্টর (physical transistors) দ্বারা এনফোর্স করা হয়।

এই হার্ডওয়্যার-লেভেলের ডিফেন্সটিই হলো **এআরএম মেমরি ট্যাগিং এক্সটেনশন বা ARM Memory Tagging Extension (MTE)**।

MTE সাইবার সিকিউরিটির জগতে একটি যুগান্তকারী প্যারাডাইম শিফট (paradigm shift)-এর প্রতিনিধিত্ব করে। কম্পিউটিংয়ের ইতিহাসে প্রথমবারের মতো সিলিকন (silicon) বা প্রসেসর নিজেই রিয়েল-টাইমে মেমরি করাপশন এক্সপ্লয়টগুলোকে (memory corruption exploits) ট্র্যাক ও কিল করার কাজে সক্রিয়ভাবে অংশগ্রহণ করে। এটি কেন এত বৈপ্লবিক তা বুঝতে হলে, প্রথমে আমাদের মেমরি করাপশন ভালনারেবিলিটির (memory corruption vulnerabilities) সবচেয়ে বিপজ্জনক দুটি ক্লাস সম্পর্কে বুঝতে হবে, যা মূলত বিশ্বের শীর্ষস্থানীয় সাইবার অ্যাটাকগুলোকে পরিচালনা করে।

দ্য এনিমি: মেমরি করাপশন (The Enemy: Memory Corruption)

আধুনিক অপারেটিং সিস্টেমগুলোর (Android, iOS, Windows এবং Linux সহ) প্রায় ৭০% ক্রিটিক্যাল এবং এক্সপ্লয়টেবল ভালনারেবিলিটি মূলত একটি নির্দিষ্ট ক্যাটাগরির অন্তর্ভুক্ত: **মেমরি সেইফটি বাগস (memory safety bugs)**। এগুলো মূলত সফটওয়্যার কীভাবে কম্পিউটারের র‍্যান্ডম অ্যাক্সেস মেমরি (RAM) পরিচালনা করে, সে সম্পর্কিত ত্রুটি। এক্সপ্লয়টের এই ল্যান্ডস্কেপে প্রধানত দুটি নির্দিষ্ট সাবটাইপ (subtype) বা উপধরন আধিপত্য বিস্তার করে।

ইউজ-আফটার-ফ্রি (Use-After-Free বা UAF)

কল্পনা করুন একটি হোটেলের কথা। একজন অতিথি রুম নম্বর ৩০৭-এ চেক-ইন করলেন এবং তাকে একটি ফিজিক্যাল কি-কার্ড (key card) দেওয়া হলো। দুই রাত থাকার পর, অতিথি চেক-আউট করে চলে গেলেন। হোটেলের উচিত সাথে সাথেই কি-কার্ডটি ডিঅ্যাক্টিভেট করা এবং পরবর্তী অতিথির জন্য রুমটিকে স্যানিটাইজ করা।

এখন কল্পনা করুন, হোটেল কর্তৃপক্ষ একটি ভুল করে বসল। তারা পুরোনো কি-কার্ডটি ডিঅ্যাক্টিভেট করতে ভুলে গেল। একজন নতুন অতিথি রুম ৩০৭-এ চেক-ইন করলেন এবং তার নিজের জিনিসপত্র—যেমন ল্যাপটপ, মানিব্যাগ, পাসপোর্ট—সেখানে রাখলেন।

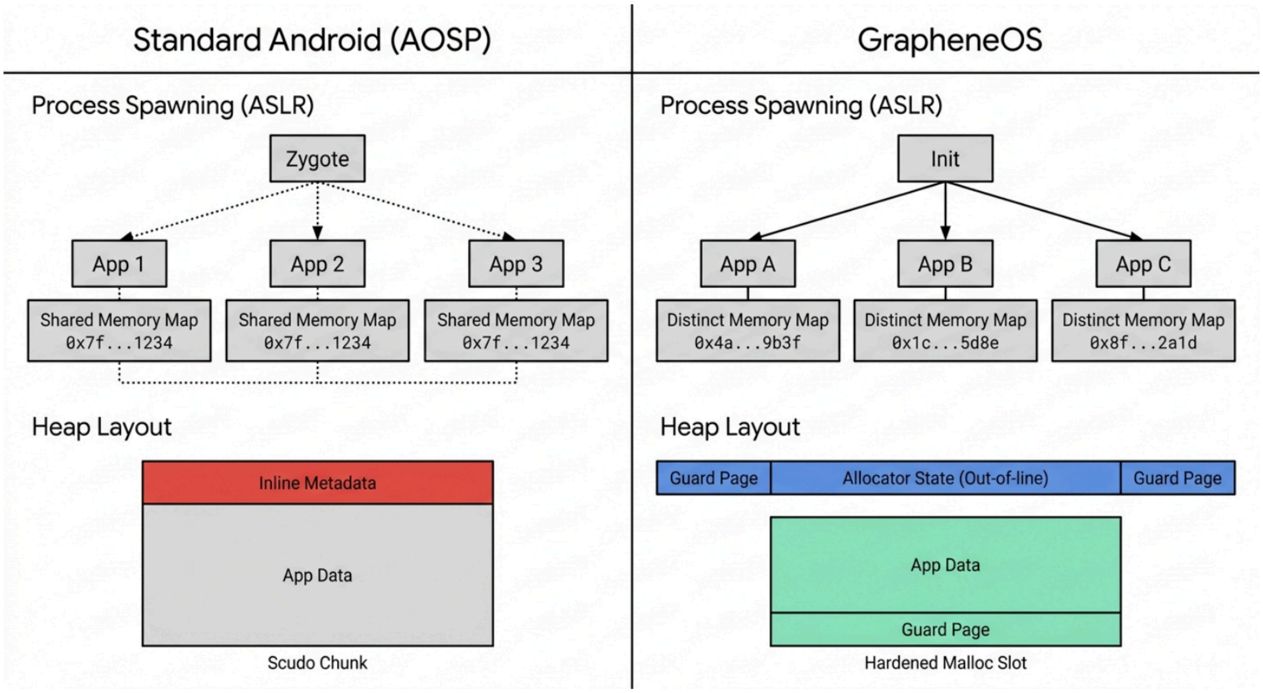
এদিকে মূল অতিথির কাছে তখনও একটি কার্যকর বা ওয়ার্কিং কি-কার্ড রয়েছে। তিনি যদি ৩০৭ নম্বর রুমে ফিরে এসে দরজা খোলেন, তবে তিনি সম্পূর্ণ বিনা বাধায় নতুন অতিথির ব্যক্তিগত জিনিসপত্রে ফুল অ্যাক্সেস পেয়ে যাবেন। তিনি ল্যাপটপ চুরি করতে পারেন, পাসপোর্ট দেখতে পারেন, বা এমনকি একটি লিসেনিং ডিভাইস বা আড়িপাতার যন্ত্রও (listening device) সেখানে লুকিয়ে রাখতে পারেন।

সফটওয়্যারের ভাষায়, এটিই হলো **ইউজ-আফটার-ফ্রি (Use-After-Free)**। একটি প্রোগ্রাম মেমরির একটি ব্লক বা অংশ অ্যালোকেট করে (হোটেলের রুম), সেটি ব্যবহার করে এবং তারপর সেটিকে "ফ্রি" (free) করে দেয় (অতিথির চেক-আউট)। মেমরিটি তখন মেমরি পুলে (pool) ফিরে যায় এবং অন্য একটি প্রক্রিয়ার বা প্রসেসের জন্য পুনরায় অ্যালোকেটেড হয় (নতুন অতিথির চেক-ইন)। তবে, মূল প্রোগ্রামটি ওই নির্দিষ্ট মেমরি ব্লকের দিকে একটি ড্যাংলিং পয়েন্টার বা dangling pointer (পুরোনো কি-কার্ড) তার কাছে রেখে দেয়। মূল প্রোগ্রামটি যদি ওই ড্যাংলিং পয়েন্টারের মাধ্যমে ফ্রি করা মেমরিটিতে অ্যাক্সেস করে, তবে এটি নতুন দখলকারীর ডেটা পড়তে বা করাপ্ট (corrupt) করতে পারে।

যে অ্যাটাকার বা আক্রমণকারী একটি UAF ভালনারেবিলিটি খুঁজে পায়, সে মেমরি অ্যালোকেটরকে স্ট্র্যাটেজিক বা কৌশলগতভাবে ম্যানিপুলেট করে সেই ফ্রি করা মেমরি ব্লকে তার নিজস্ব ম্যালিসিয়াস ডেটা প্লেস করতে পারে। যখন সেই ড্যাংলিং পয়েন্টারটি ডিরেফারেন্সড (dereferenced) হয়, তখন প্রোগ্রামটি নিজের অজান্তেই অ্যাটাকারের পেলোড এক্সিকিউট করে ফেলে।

আউট-অফ-বাউন্ডস (Out-of-Bounds বা OOB) অ্যাক্সেস

Architectural Divergence in Memory Safety Models



Standard Android utilizes Scudo with inline metadata and Zygote forking, leading to shared ASLR across applications. GrapheneOS utilizes out-of-line metadata isolation, 48-bit ASLR via exec spawning, and hardware-enforced MTE.

হোটেলের অ্যানালজি বা রূপকটিতে আবার ফিরে যাওয়া যাক। এবার, ৩০৭ নম্বর রুমের অতিথিকে এমন একটি কি-কার্ড দেওয়া হলো যা দিয়ে শুধুমাত্র ৩০৭ নম্বর রুমটিই খোলার কথা। কিন্তু, হোটেলের ইলেকট্রনিক লক সিস্টেমের একটি প্রোগ্রামিং ত্রুটির কারণে, কি-কার্ডটি আসলে ৩০৭ থেকে ৩১২ নম্বর পর্যন্ত সমস্ত রুম খুলতে সক্ষম।

অতিথি করিডোর দিয়ে হেঁটে গিয়ে ৩১০ নম্বর রুমটি খুললেন (যা সম্পূর্ণ ভিন্ন একজন এবং সন্দেহমুক্ত অতিথির) এবং তার জিনিসপত্র ঘাঁটাঘাঁটি শুরু করলেন।

সফটওয়্যারের ভাষায়, এটি একটি **আউট-অফ-বাউন্ডস (Out-of-Bounds)** অ্যাক্সেস। একটি প্রোগ্রামকে মেমরির একটি নির্দিষ্ট ব্লক (ধরা যাক, ৬৪ বাইট) অ্যালোকেট করা হয়, কিন্তু কোডিংয়ে থাকা কোনো এরর বা ত্রুটির কারণে (সাধারণত বাফার ওভারফ্লো বা একটি ইনটিজার মিসক্যালকুলেশন), প্রোগ্রামটি তার অ্যালোকেটেড রিজিওনের বাউন্ডারির জ্ঞ বাইরে গিয়ে রিড (read) বা রাইট (write) করে ফেলে। এর ফলে এটি একটি ভিন্ন ডেটা স্ট্রাকচার বা প্রসেসের সংলগ্ন মেমরিতে (adjacent memory) অ্যাক্সেস পেয়ে যায়।

একটি OOB ভালনারেবিলিটি এক্সপ্লয়েট করা অ্যাটাকার, পাশের মেমরিতে থাকা ক্রিটিক্যাল ডেটা স্ট্রাকচারগুলো—যেমন ফাংশন পয়েন্টার (function pointers) বা রিটার্ন অ্যাড্রেস (return addresses)—ওভাররাইট করতে পারে। এতে করে সে প্রোগ্রামের এক্সিকিউশন ফ্লো বা ফ্লো অব এক্সিকিউশন (execution flow) হাইজ্যাক করে ফেলে এবং এটিকে অ্যাটাকার-নিয়ন্ত্রিত শেলকোডের (shellcode) দিকে রিডিরেক্ট করে দেয়।

দ্য হার্ডওয়্যার সল্যুশন: কালারড স্টিকি নোটস (The Hardware Solution: Colored Sticky Notes)

`hardened_malloc`-এর মতো সফটওয়্যার-ভিত্তিক ডিফেন্সগুলো এসব অ্যাটাক বা আক্রমণ মিটিগেট (mitigate) করার ক্ষেত্রে উল্লেখযোগ্যভাবে কার্যকর, তবে এগুলো সিপিইউ-এর বিদ্যমান ইনস্ট্রাকশন সেটের (instruction set) সীমাবদ্ধতার ভেতরে থেকেই কাজ করে। এরা চেক যুক্ত করে, মেমরি লেআউট র‍্যান্ডমাইজ করে এবং গার্ড পেইজ (guard pages) ইনসার্ট বা প্রবেশ করায়। তবে, মেমরি অ্যালোকেটরের অভ্যন্তরীণ কাঠামো সম্পর্কে গভীরভাবে অবগত একজন অত্যন্ত দক্ষ অ্যাটাকার তাত্ত্বিকভাবে এমন একটি এক্সপ্লয়েট তৈরি করতে পারে যা এই সফটওয়্যার-লেভেলের ডিফেন্সগুলোকে ফাঁকি দিতে সক্ষম।

ARM মেমরি ট্যাগিং এক্সটেনশন এই থিওরিটিক্যাল গ্যাপ বা তাত্ত্বিক শূন্যস্থানটিকে দূর করে দেয় ডিফেন্স ব্যবস্থাকে সরাসরি সিলিকনে (silicon) স্থানান্তরিত করার মাধ্যমে।

MTE অত্যন্ত মার্জিত এবং সহজ একটি নীতির ওপর কাজ করে। এটিকে এভাবে চিন্তা করা যেতে পারে যে, সিপিইউ শারীরিকভাবে বা ফিজিক্যালি অ্যালোকেট করা প্রতিটি মেমরি ব্লকের সাথে একটি **কালারড স্টিকি নোট (colored sticky note)** বা রঙিন লেবেল লাগিয়ে দেয়।

যখন `hardened_malloc` একটি ইমেজ রেন্ডারিং প্রসেসের জন্য ৬৪-বাইট বাফার (64-byte buffer) অ্যালোকেট করে, তখন সিপিইউ সেই বাফারের জন্য একটি র‍্যান্ডম ৪-বিট ট্যাগ (4-bit tag) বা ১৬টি সম্ভাব্য রঙের একটি অ্যাসাইন করে। এই ট্যাগটি একটি ডেডিকেটেড হার্ডওয়্যার রেজিস্টারে সংরক্ষিত থাকে এবং শারীরিকভাবে মেমরি অ্যাড্রেসটির সাথে যুক্ত থাকে। একই সময়ে, পয়েন্টারটিকেও (যে ভেরিয়েবলটি প্রোগ্রামটি বাফার অ্যাক্সেস করতে ব্যবহার করে) একই রঙ দিয়ে ট্যাগ করা হয়।

যখন প্রোগ্রামটি পয়েন্টারের মাধ্যমে মেমরিতে অ্যাক্সেস করার চেষ্টা করে, তখন সিপিইউ একটি তাৎক্ষণিক এবং জিরো-কস্ট (zero-cost) হার্ডওয়্যার চেক সম্পন্ন করে: **পয়েন্টারের রঙ কি মেমরির রঙের সাথে মেলে?**

উত্তর যদি হ্যাঁ হয়, তবে অ্যাক্সেসটি ফুল স্পিডে চলতে থাকে। এতে কোনো পারফরম্যান্স পেনাল্টি (performance penalty) বা গতির অবনতি ঘটে না।

আর উত্তর যদি না হয়, তবে সিপিইউ একটি তাৎক্ষণিক এবং আনরিকভারেবল হার্ডওয়্যার ফল্ট (unrecoverable hardware fault) বা ত্রুটি রেইজ করে। প্রসেসটি সাথে সাথে টার্মিনেট বা বন্ধ হয়ে যায়। কোনো এক্সেপশন হ্যান্ডলার (exception handler) থাকে না। কোনো গ্রেসফুল ডিগ্রেডেশন (graceful degradation) বা ধীরে ধীরে প্রসেস বন্ধ হওয়ার সুযোগ থাকে না। প্রোগ্রামটিকে ঠিক ন্যানোসেকেন্ডের মধ্যে কিল (killed) বা ধ্বংস করা হয়।

এমটিই বনাম ইউজ-আফটার-ফ্রি (MTE vs. Use-After-Free)

MTE এনাবল করা অবস্থায় হোটেলের অ্যানালজিটি (analogy) পুনরায় দেখা যাক।

মূল অতিথি ৩০৭ নম্বর রুমে চেক-ইন করে একটি **নীল (blue)** কি-কার্ড পান। রুমটিকেও **নীল** রঙে ট্যাগ করা হয়। যখন অতিথি চেক-আউট করেন, তখন হোটেল শুধু কি-কার্ডটি ডিঅ্যাক্টিভেট করে না—তারা রুমটিতে ফিজিক্যালি নতুন করে রং করে দেয়। ফলে, রুম ৩০৭ এখন **লাল (red)** ট্যাগযুক্ত।

একজন নতুন অতিথি চেক-ইন করেন এবং একটি **লাল** কি-কার্ড পান। সব কিছুই নিখুঁতভাবে কাজ করে।

মূল অতিথি তার পুরোনো **নীল** কি-কার্ড নিয়ে ফিরে আসেন এবং ৩০৭ নম্বর রুমের লকে সোয়াইপ করেন। লকটি কার্ডের রঙের (নীল) সাথে রুমের রঙের (লাল) তুলনা করে। **মিসম্যাচ ডিটেক্টেড (Mismatch detected)।** লকটি খুলতে অস্বীকার করে এবং একটি অ্যালার্ম ট্রিগার করে।

MTE-এর পরিভাষায়, যখন মেমরি ফ্রি করা হয় এবং পরবর্তীতে সেটি রিঅ্যালোকেট (reallocate) করা হয়, তখন `hardened_malloc` রিঅ্যালোকেটেড রিজিওনের জন্য একটি নতুন র‍্যান্ডম ট্যাগ অ্যাসাইন করে। কিন্তু ড্যাংলিং পয়েন্টারটি তখনও পুরোনো ট্যাগটি বহন করে। ড্যাংলিং পয়েন্টারটি রিঅ্যালোকেটেড মেমরিতে অ্যাক্সেস করার চেষ্টা করা মাত্রই, সিপিইউ ট্যাগ মিসম্যাচ (tag mismatch) শনাক্ত করে এবং তৎক্ষণাৎ প্রসেসটি টার্মিনেট করে দেয়। হার্ডওয়্যারের মাধ্যমে ইউজ-আফটার-ফ্রি (Use-After-Free) এক্সপ্লয়টটি আনঅথরাইজড (unauthorized) ডেটার একটি সিঙ্গেল বাইট পড়ার আগেই কিল হয়ে যায়।

এমটিই বনাম আউট-অফ-বাউন্ডস অ্যাক্সেস (MTE vs. Out-of-Bounds Access)

এবার OOB দৃশ্যপটের কথা বিবেচনা করুন। রুম ৩০৭-কে **সবুজ (green)** ট্যাগ করা হয়েছে। রুম ৩০৮-কে **হলুদ (yellow)** ট্যাগ করা হয়েছে। রুম ৩০৭-এর অতিথির কাছে একটি **সবুজ** কি-কার্ড রয়েছে।

সফটওয়্যারের ত্রুটির কারণে, অতিথির প্রোগ্রামটি রুম ৩০৮-এ অ্যাক্সেস করার চেষ্টা করে। সিপিইউ পয়েন্টারের ট্যাগটির (**সবুজ**) সাথে রুম ৩০৮-এর মেমরি ট্যাগের (**হলুদ**) তুলনা করে। **মিসম্যাচ (Mismatch)।** হার্ডওয়্যার ফল্ট। প্রসেস টার্মিনেটেড (Process terminated)।

বাফার ওভারফ্লো কখনোই পাশের মেমরি পর্যন্ত পৌঁছাতে পারে না। অ্যাটাকারের পেলোড কখনোই পার্শ্ববর্তী ডেটা স্ট্রাকচারকে করাপ্ট বা ধ্বংস করতে পারে না। এক্সপ্লয়টটি সেখানেই শেষ হয়ে যায়।

GrapheneOS-এ এমটিই: ইমপ্লিমেন্টেশন (MTE on GrapheneOS: The Implementation)

সমগ্র অপারেটিং সিস্টেম জুড়ে MTE অ্যাডপ্ট বা গ্রহণ করার ক্ষেত্রে স্ট্যান্ডার্ড Android বেশ ধীরগতিসম্পন্ন। স্টক পিক্সেল ফার্মওয়্যারে (stock Pixel firmware), MTE শুধুমাত্র সিস্টেম প্রসেসের একটি সীমিত সাবসেটের জন্য এনাবল করা থাকে, যা অপারেটিং সিস্টেমের বিশাল একটি অংশকে আনপ্রোটেক্টেড (unprotected) বা অরক্ষিত রাখে।

GrapheneOS এই আপস্ট্রিম ডিফল্টের (upstream default) বাইরে গিয়ে অত্যন্ত অ্যাগ্রেসিভভাবে (aggressively) MTE কভারেজকে প্রসারিত করে। এটি সিস্টেম প্রসেস, নেটিভ ডিমন (native daemons) এবং মিডিয়া কোডেক স্ট্যাক, ব্লুটুথ স্ট্যাক এবং ভ্যানাডিয়াম (Vanadium) ব্রাউজার রেন্ডারারের মতো ক্রিটিক্যাল অ্যাটাক সারফেস কম্পোনেন্টগুলোর (critical attack surface components) একটি বিস্তৃত সেটের উপর **সিঙ্ক্রোনাস মোডে বা synchronous mode**-এ (সবচেয়ে কঠোর এনফোর্সমেন্ট, যেখানে ট্যাগ মিসম্যাচের কারণে একটি তাৎক্ষণিক ও ডিটারমিনিস্টিক ক্র্যাশ ঘটে) MTE এনাবল করে।

`hardened_malloc`-এর সফটওয়্যার-লেভেলের গার্ড পেইজ এবং র‍্যান্ডমাইজেশনের সাথে MTE-এর হার্ডওয়্যার-লেভেলের ট্যাগ এনফোর্সমেন্টকে যুক্ত করে, GrapheneOS এমন একটি ডিফেন্স-ইন-ডেপ্‌থ (defense-in-depth) তৈরি করে যেখানে হার্ডওয়্যারের মিস করা ত্রুটিগুলো সফটওয়্যার ধরে ফেলে এবং সফটওয়্যারের মিস করা ত্রুটিগুলো হার্ডওয়্যার ধরে ফেলে।

এর ফলাফল হলো এমন একটি মেমরি সেইফটি পসচার (memory safety posture), যা মোবাইল ইন্ডাস্ট্রিতে গাণিতিকভাবেই নজিরবিহীন। Pixel 8 বা এর পরবর্তী হার্ডওয়্যারে চলা GrapheneOS ডিভাইসে মেমরি করাপশন ভালনারেবিলিটি এক্সপ্লয়ট করার চেষ্টা করা কোনো অ্যাটাকার শুধু সফটওয়্যার চেকের সাথেই লড়াই করে না—তারা সরাসরি সিলিকনের নিয়মের সাথেই লড়াই করে। ব্লু টিমের (Blue Team) হয়ে সিপিইউ নিজেই এখানে একজন সক্রিয় যোদ্ধা বা কমব্যাট্যান্ট (combatant) হিসেবে কাজ করে।

কালারড স্টিকি নোটগুলো সদা সতর্ক নজর রাখছে। এবং তারা কখনোই তাদের চোখের পলক ফেলে না।

৩.৬ ডিপ ডাইভ: hardened_malloc অ্যালোকেটর

পূর্ববর্তী সেকশনগুলোতে আমরা বারবার `hardened_malloc`-কে একজন অদৃশ্য রক্ষক হিসেবে উল্লেখ করেছি, যা মেমরি করাপশন এক্সপ্লয়টগুলো (memory corruption exploits) এক্সিকিউট বা কার্যকরী হওয়ার আগেই সেগুলোকে টার্মিনেট বা নিষ্ক্রিয় করে দেয়। আমরা দেখেছি কীভাবে এটি বাফার ওভারফ্লো (buffer overflows) ধরে ফেলে, পেগাসাস-ক্লাস পেলোডগুলো (Pegasus-class payloads) নিউট্রালাইজ করে এবং ফ্রি করা মেমরি পেইজগুলো স্যানিটাইজ করে। এখন সময় হয়েছে এর হুডটি খুলে ইঞ্জিনের ভেতরের কাজ পরীক্ষা করে দেখার।

`hardened_malloc` কীভাবে কাজ করে তা বুঝতে হলে, প্রথমেই আপনাকে বুঝতে হবে যে এটি মূলত কোন কম্পোনেন্ট বা উপাদানটির জায়গা দখল করে কাজ করছে: আর তা হলো মেমরি অ্যালোকেটর (memory allocator)।

দ্য রোল অব দ্য মেমরি অ্যালোকেটর (The Role of the Memory Allocator)

আপনার ডিভাইসে চলমান প্রতিটি অ্যাপ্লিকেশন—আপনার ব্রাউজার, আপনার মেসেজিং অ্যাপ, আপনার ক্যামেরা—এদের অপারেটিং সিস্টেম থেকে প্রতিনিয়ত RAM-এর ব্লক ধার করার প্রয়োজন হয়। যখন Vanadium একটি ওয়েব পেইজ লোড করে, তখন পেইজটির HTML স্ট্রাকচার, এর ছবি, JavaScript অবজেক্ট এবং রেন্ডারিং বাফারগুলো স্টোর বা সংরক্ষণ করার জন্য এটি মেমরির রিকুয়েস্ট বা অনুরোধ করে। আপনি যখন ট্যাবটি বন্ধ করে দেন, তখন সেই মেমরিটি অন্যান্য অ্যাপ্লিকেশনগুলোর ব্যবহারের জন্য পুলে (pool) ফেরত পাঠাতে হয়।

মেমরি অ্যালোকেটর হলো সিস্টেম-লেভেলের সেই লাইব্রেরিয়ান যা এই পুরো প্রক্রিয়াটি ম্যানেজ বা পরিচালনা করে। প্রতিটি অ্যাপ্লিকেশনের ডেটা RAM-এর কোথায় রাখা হবে, প্রতিটি অ্যালোকেট করা ব্লকের সাইজ কতটা হবে এবং কাজ শেষে ব্লকটি ফেরত দেওয়ার পর সেটির সাথে কী করা হবে—তা এই মেমরি অ্যালোকেটরই নির্ধারণ করে।

স্ট্যান্ডার্ড কমার্শিয়াল Android-এ, ডিফল্ট মেমরি অ্যালোকেটরটির নাম হলো **Scudo**। Scudo একটি অত্যন্ত কার্যকর এবং জেনারেল-পারপাস অ্যালোকেটর, যা প্রাথমিকভাবে পারফরম্যান্সের দিকে লক্ষ্য রেখে ডিজাইন করা হয়েছে। এটি মেমরি ব্লকগুলোকে খুব দ্রুত এবং দক্ষতার সাথে ডিস্ট্রিবিউট বা বন্টন করে দেয় এবং এতে বেসিক কিছু সিকিউরিটি চেক অন্তর্ভুক্ত রয়েছে, যা সবচেয়ে সুস্পষ্ট মেমরি করাপশন বাগগুলো (memory corruption bugs) শনাক্ত করতে পারে।

তবে, Scudo-কে মূলত স্পিড বা গতি এবং সিকিউরিটি বা নিরাপত্তার মধ্যে একটি ব্যালেন্স বা ভারসাম্য বজায় রাখার জন্য তৈরি করা হয়েছিল। এর ফলে এতে এমন কিছু আর্কিটেকচারাল কম্প্রোমাইজ (architectural compromises) বা স্থাপত্যগত আপোস করা হয়েছে, যা সুযোগ বুঝে অত্যন্ত দক্ষ অ্যাটাকার বা আক্রমণকারীরা সহজেই এক্সপ্লয়ট বা কাজে লাগাতে পারে।

স্ট্যান্ডার্ড অ্যালোকেটরের আর্কিটেকচারাল উইকনেস বা দুর্বলতা (The Architectural Weaknesses of Standard Allocators)

Scudo-এর মতো স্ট্যান্ডার্ড অ্যালোকেটরগুলো মূলত তিনটি ফান্ডামেন্টাল বা মৌলিক দুর্বলতায় ভুগে থাকে, যেগুলোর সুযোগ এলিট রেড টিমগুলো (Red Teams) প্রতিনিয়তই নিয়ে থাকে:

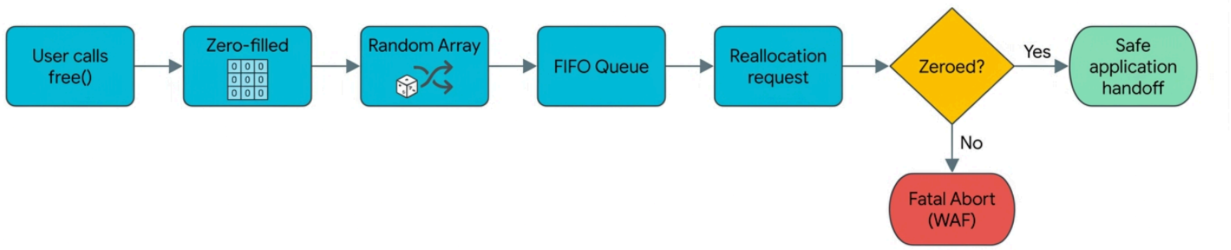
১. প্রেডিক্টেবল হিপ লেআউট (Predictable Heap Layout): Scudo যখন মেমরি অ্যালোকেট করে, তখন ব্লকগুলোকে একটি তুলনামূলকভাবে প্রেডিক্টেবল বা অনুমানযোগ্য অর্ডারে বা ক্রমানুসারে স্থাপন করা হয়। যদি কোনো অ্যাটাকার জানে যে মেমরিতে বাফার A-কে সর্বদা বাফার B-এর ঠিক আগে স্থাপন করা হয়, তবে তারা সুনির্দিষ্টভাবে বাফার B-কে করাপ্ট করার জন্য ইচ্ছাকৃতভাবে বাফার A-কে ওভারফ্লো করাতে পারে। এই প্রেডিক্টেবল বা অনুমানযোগ্য দিকটিই হলো **হিপ গ্রুমিং (heap grooming)** নামক একটি বিশেষ টেকনিকের মূল ভিত্তি—যেখানে অ্যাটাকার মূলত অ্যালোকেটরকে ম্যানিপুলেট করে তাদের ম্যালিসিয়াস ডেটাকে একটি নির্দিষ্ট এবং এক্সপ্লয়টেবল বা ব্যবহারযোগ্য বিন্যাসে স্থাপন করতে বাধ্য করে।

২. ইনলাইন মেটাডেটা (Inline Metadata): Scudo এর নিজস্ব বুককিপিং ডেটা বা ইন্টার্নাল হিসাব-নিকাশ (মেটাডেটা বা metadata)—যেমন একটি ব্লকের সাইজ এবং এর অ্যালোকেশন স্ট্যাটাস—একই মেমরি রিজিয়নে বা অঞ্চলে ইউজার ডেটার ঠিক পাশেই সংরক্ষণ করে। যদি কোনো অ্যাটাকার একটি বাফার ওভারফ্লো করে, তবে তারা এই সংলগ্ন মেটাডেটাটিকে করাপ্ট করতে পারে। এই করাপ্ট হওয়া মেটাডেটাটিকে খুব সতর্কতার সাথে ক্র্যাফট করার মাধ্যমে, অ্যাটাকার অ্যালোকেটরকে ধোঁকা দিয়ে পরবর্তী অ্যালোকেশন বা ডিঅ্যালোকেশন অপারেশনের সময় অ্যাটাকার-নিয়ন্ত্রিত ডেটাকে যেকোনো একটি মেমরি অ্যাড্রেসে লিখে ফেলতে পারে। এটিই মূলত অত্যন্ত ধ্বংসাত্মক **হিপ মেটাডেটা করাপশন (heap metadata corruption)** অ্যাটাক ক্লাসের প্রধান ভিত্তি।

৩. উইক ফ্রি-লিস্ট ম্যানেজমেন্ট (Weak Free-List Management): মেমরি যখন ফ্রি করা হয়, তখন ভবিষ্যতে পুনরায় ব্যবহারের জন্য Scudo সেই ফ্রি করা ব্লকটিকে একটি ইন্টার্নাল লিস্টে বা অভ্যন্তরীণ তালিকায় ("free list") রেখে দেয়। স্ট্যান্ডার্ড অ্যালোকেটরগুলো প্রায়শই লাস্ট-ইন-ফার্স্ট-আউট (LIFO) অর্ডারে ফ্রি করা ব্লকগুলোকে পুনরায় ব্যবহার করে। এই প্রেডিক্টেবল রিইউজ প্যাটার্ন বা পুনর্ব্যবহারযোগ্য কাঠামোটিই ইউজ-আফটার-ফ্রি (Use-After-Free) এক্সপ্লয়টগুলোকে অভাবনীয়ভাবে রিলায়েবল বা নির্ভরযোগ্য করে তোলে—একজন অ্যাটাকার একটি ব্লক ফ্রি করে, সাথে সাথে একই সাইজের একটি নতুন ব্লক অ্যালোকেট করে, এবং অ্যালোকেটর তাকে ঠিক ওই একই মেমরি রিজিয়ন বা অঞ্চলটি দিয়ে দেয়, যা ড্যাংলিং পয়েন্টার (dangling pointer) এক্সপ্লয়টেশনের জন্য একেবারে নিখুঁত একটি ক্ষেত্র প্রস্তুত করে।

GrapheneOS-এর রেসপন্স: hardened_malloc (The GrapheneOS Response: hardened_malloc)

Mitigating Temporal Exploits: The hardened_malloc Quarantine Flow



Freed memory undergoes immediate zeroing before entering a non-deterministic two-stage quarantine. Upon reallocation, the allocator verifies the memory remains perfectly zeroed, neutralizing Write-After-Free exploits.

GrapheneOS Scudo-কে রিপ্লেস করে `hardened_malloc` ব্যবহার করে—এটি মূলত GrapheneOS প্রজেক্ট দ্বারা গোড়া থেকে তৈরি করা একটি কাস্টম এবং সম্পূর্ণ সিকিউরিটি-ফোকাসড মেমরি অ্যালোকেটর। এটি Scudo-এর কোনো প্যাচড ভার্সন বা সংশোধিত সংস্করণ নয়। এটি এমন একটি মৌলিকভাবে ভিন্ন আর্কিটেকচার বা স্থাপত্য, যা উপরে বর্ণিত প্রতিটি দুর্বলতাকে হয় সম্পূর্ণ অসম্ভব, নতুবা সম্ভাবনার দিক থেকে একেবারে অবাস্তব করে তোলার জন্য বিশেষভাবে ইঞ্জিনিয়ারিং করা হয়েছে।

১. র‍্যান্ডমাইজড হিপ লেআউট (Randomized Heap Layout): `hardened_malloc` অত্যন্ত অ্যাগ্রেসিভভাবে বা জোরালোভাবে মেমরি অ্যালোকেশনের স্থানগুলোকে র‍্যান্ডমাইজ বা এলোমেলো করে দেয়। যখন কোনো অ্যাপ্লিকেশন একই সাইজের দুটি বাফারের রিকুয়েস্ট করে, তখন `hardened_malloc` তাদের প্রেডিক্টেবলভাবে বা অনুমানের ভিত্তিতে

পাশাপাশি রাখে না। বরং এটি ক্রিপ্টোগ্রাফিক র‍্যান্ডমাইজেশন (cryptographic randomization) ব্যবহার করে পুরো খালি মেমরি স্পেস জুড়ে সেগুলোকে ছড়িয়ে ছিটিয়ে দেয়।

এটি হিপ গ্রুমিং (heap grooming) টেকনিকটিকে একেবারে ধ্বংস করে দেয়। বাফার A ঠিক বাফার B-এর কোথায় বা কত দূরে অবস্থিত, রেড টিম (Red Team) আর কখনোই তা প্রেডিক্ট বা অনুমান করতে পারে না। তাদের ওভারফ্লো পেলোডটি এমন একটি র‍্যান্ডমাইজড বা এলোমেলো এবং আনপ্রেডিক্টেবল বা অপ্রত্যাশিত রিজিয়নে গিয়ে ল্যান্ড করে—যা খুব সম্ভবত নির্দিষ্ট কোনো টার্গেটকে নিখুঁতভাবে করাপ্ট করার বদলে একটি গার্ড পেইজের (guard page) ওপর আছড়ে পড়ে কোনো রকম ক্ষতি ছাড়াই ক্র্যাশ করে।

২. আইসোলেটেড মেটাডেটা (Isolated Metadata): `hardened_malloc` ফিজিক্যালি সমস্ত অ্যালোকেটর মেটাডেটাকে ইউজার ডেটা থেকে সেপারেট বা আলাদা করে রাখে। এর ইন্টার্নাল বুককিপিং স্ট্রাকচারগুলো বা অভ্যন্তরীণ হিসাব-নিকাশের তথ্যগুলো সম্পূর্ণ ভিন্ন একটি ডেডিকেটেড মেমরি রিজিয়নে সংরক্ষিত থাকে, যেখানে ইউজার-ফেসিং বাফারগুলো (user-facing buffers) কোনোভাবেই অ্যাক্সেস করতে পারেবিধা পায় না।

ফলে, কোনো অ্যাটাকার যদি একটি বাফার ওভারফ্লো করেও থাকে, তবুও সে কখনোই কোনো অ্যালোকেটর মেটাডেটার সন্ধান পাবে না। অর্থাৎ, সেখানে করাপ্ট করার মতো কিছুই থাকে না। এর ফলে হিপ মেটাডেটা করাপশন (heap metadata corruption) অ্যাটাক ক্লাসটি শুধু মিটিগেট বা কমানোই হয় না, বরং আর্কিটেকচারালি বা স্থাপত্যগতভাবে একে চিরতরে দূর করে দেওয়া হয়।

৩. স্ট্রিক্ট সাইজ-ক্লাস সেগ্রিগেশন (Strict Size-Class Segregation): `hardened_malloc` পুরো হিপটিকে কঠোরভাবে সেগ্রিগেটেড বা পৃথক সাইজ ক্লাসে অর্গানাইজ বা সাজিয়ে রাখে। ১৬ বাইটের অ্যালোকেশনগুলোকে একটি ডেডিকেটেড বা নির্দিষ্ট ১৬-বাইটের পুলে রাখা হয়। ৬৪ বাইটের অ্যালোকেশনগুলোকে সম্পূর্ণ আলাদা একটি ৬৪-বাইটের পুলে রাখা হয়। ঠিক একইভাবে ২৫৬ বাইটের অ্যালোকেশনগুলো চলে যায় নিজস্ব একটি পুলে।

এই ডিজাইনের একটি অত্যন্ত গভীর সিকিউরিটি ইমপ্লিকেশন বা প্রভাব রয়েছে। একটি স্ট্যান্ডার্ড অ্যালোকেটরের ক্ষেত্রে, একজন অ্যাটাকার চাইলেই একটি ৬৪-বাইট অবজেক্টকে ফ্রি করার পর তাদের ম্যালিসিয়াস পেলোডযুক্ত একটি ৬৪-বাইট বাফার অ্যালোকেট করতে পারে। কারণ তারা আগে থেকেই জানে যে অ্যালোকেটর তাদের ঠিক একই মেমরি রিজিয়নটিই দেবে (যার ফলে ইউজ-আফটার-ফ্রি বা Use-After-Free সংঘটিত হয়)। কিন্তু কঠোর সাইজ-ক্লাস সেগ্রিগেশন এবং প্রতিটি ক্লাসের ভেতরে র‍্যান্ডমাইজড স্লট সিলেকশনের (randomized slot selection) কারণে, এই ডিটারমিনিস্টিক রিইউজ বা নিশ্চিত পুনর্ব্যবহারযোগ্যতা একদম দূর হয়ে যায়। পুলের ভেতরের ঠিক কোন স্লটটি অ্যাসাইন করা হবে, তা অ্যাটাকার আর কোনোভাবেই নির্ভরযোগ্যতার সাথে প্রেডিক্ট বা অনুমান করতে পারে না।

৪. গার্ড পেইজেস (Guard Pages): `hardened_malloc` মেমরি রিজিয়নের সীমানায় বা বাউন্ডারিতে আনম্যাপড **গার্ড পেইজ (guard pages)** ইনসার্ট বা যুক্ত করে। একটি গার্ড পেইজ হলো মূলত মেমরির এমন একটি ব্লক, যেটিকে কার্নেল (kernel) ইচ্ছাকৃতভাবে সম্পূর্ণ ইনঅ্যাক্সেসিবল বা অগম্য হিসেবে মার্ক করে রাখে। যেকোনো রিড বা রাইট অপারেশন যা একটি গার্ড পেইজকে স্পর্শ করে, তা তৎক্ষণাৎ একটি আনরিকভারেবল হার্ডওয়্যার ফল্ট বা পুনরায় ঠিক করা যায় না এমন ত্রুটি ট্রিগার করে।

তাই যখনই কোনো বাফার ওভারফ্লো ঘটে, ওভারফ্লো হওয়া ডেটা সরাসরি গিয়ে সেই গার্ড পেইজে আঘাত হানে। সাথে সাথেই হার্ডওয়্যার ফল্টটি ফায়ার করে বা জ্বলে ওঠে। প্রসেসটি টার্মিনেট বা বন্ধ হয়ে যায় এবং এক্সপ্লয়টটি সেখানেই মারা যায়।

সিস্টেম-ওয়াইড ডিপ্লয়মেন্ট (System-Wide Deployment)

স্ট্যান্ডার্ড Android-এর ক্ষেত্রে, কোনো ডেভেলপার যদি তার নিজস্ব অ্যাপ্লিকেশনের ভেতরে অত্যন্ত নিষ্ঠার সাথে একটি সিকিউর অ্যালোকেটর ব্যবহারও করে, তবুও সিস্টেম লাইব্রেরি, ব্যাকগ্রাউন্ড ডিমন (background daemons) এবং মিডিয়া কোডেকগুলো সেই ডিফল্ট এবং দুর্বল অ্যালোকেটরটিই ব্যবহার করতে থাকে। Scudo রান করা মাত্র একটি দুর্বল কম্পোনেন্ট বা উপাদানই পুরো ডিভাইসটিকে কম্প্রোমাইজ করে ফেলার জন্য যথেষ্ট।

GrapheneOS এই গ্যাপ বা ফাঁকটুকুও দূর করে দেয় **সিস্টেম-ওয়াইড ডিফল্ট অ্যালোকেটর** হিসেবে `hardened_malloc`-কে ডিপ্লয় বা মোতায়েন করার মাধ্যমে। ডিভাইসের ভেতরের প্রতিটি সিঙ্গেল প্রসেস—ভ্যানাডিয়াম (Vanadium) ব্রাউজার থেকে শুরু করে ব্লুটুথ ডিমন, মিডিয়া কোডেক থেকে শুরু করে কার্নেলের ইউজারস্পেস হেল্পার (userspace helpers) পর্যন্ত—সবাই স্বয়ংক্রিয়ভাবে `hardened_malloc`-এর সুরক্ষাগুলো পেয়ে যায়।

আপনাকে এটি কনফিগার করতে হবে না। প্রতিটি অ্যাপের জন্য আলাদাভাবে এনাবল বা চালু করারও কোনো প্রয়োজন নেই। GrapheneOS বুট বা চালু হওয়ার সাথে সাথেই, আপনার ইনস্টল করা প্রতিটি অ্যাপ্লিকেশন, চলা প্রতিটি সিস্টেম সার্ভিস এবং আনট্রাস্টেড বা অবিশ্বস্ত ইনপুট প্রসেস করা প্রতিটি ব্যাকগ্রাউন্ড ডিমন স্বয়ংক্রিয়ভাবে `hardened_malloc`-এর র‍্যান্ডমাইজড, গার্ডেড এবং মেটাডেটা-আইসোলেটেড আর্কিটেকচারের (metadata-isolated architecture) সুরক্ষার চাদরে মোড়ানো থাকে।

এটি ঠিক দুর্গের শুধু একটি দরজায় তালা লাগানো এবং একই সাথে পুরো দুর্গের প্রতিটি দেওয়াল, মেঝে ও ছাদকে একযোগে মজবুত করার মধ্যে যে পার্থক্য, ঠিক তেমনই। `hardened_malloc` শুধু একটি নির্দিষ্ট অ্যাপ্লিকেশনকেই রক্ষা করে না; এটি সমগ্র অপারেটিং সিস্টেম জুড়ে মেমরির নিজস্ব ফিজিক্স বা পদার্থবিজ্ঞানটিকেই আরও বেশি শক্তিশালী বা হার্ডেন (harden) করে তোলে।

৩.৭ দৃশ্যকল্প ৪ (Scenario 4): MitM এবং Baseband Interception

Threat Classification: Network Exploitation / Radio Frequency Manipulation

Red Team Assets: IMSI Catcher (Stingray), Evil Twin Wi-Fi access point, DNS spoofing toolkit, SSL stripping proxy

Blue Team Assets: GrapheneOS Fortress Blueprint (LTE-Only mode, IOMMU isolation, WireGuard VPN Kill Switch, Private DNS, Per-Connection MAC Randomization)

Objective (Red Team): টার্গেটের নেটওয়ার্ক ট্রাফিক ইন্টারসেপ্ট করা, ক্রেডেনশিয়াল (credentials) হার্ভেস্ট করা এবং ফিজিক্যাল লোকেশন ট্র্যাক করা

Objective (Blue Team): ইন্টারসেপশন প্রতিহত করা। হোস্টাইল (hostile) বা বৈরী নেটওয়ার্কে ক্রিপ্টোগ্রাফিক ইনভিজিবিলিটি (cryptographic invisibility) বজায় রাখা।

অপারেশনাল কনটেক্সট (Operational Context)

টার্গেট—একজন কর্পোরেট এক্সিকিউটিভ—একটি বড় কনভেনশন সেন্টারে তিন দিনব্যাপী প্রযুক্তি সম্মেলনে অংশ নিচ্ছেন। প্রতিদ্বন্দ্বী একটি কর্পোরেশন রেড টিমকে (Red Team) নিযুক্ত করেছে এক্সিকিউটিভের কমিউনিকেশন ইন্টারসেপ্ট করতে এবং সেনসিটিভ বিজনেস ইন্টেলিজেন্স এক্সট্র্যাক্ট করতে।

কনভেনশন সেন্টারটি একটি আদর্শ হান্টিং গ্রাউন্ড (hunting ground)। হাজার হাজার ডিভাইস সক্রিয়ভাবে Wi-Fi কানেকশন খুঁজছে। সেলুলার ইনফ্রাস্ট্রাকচার অত্যন্ত কনজেস্টেড (congested), যার কারণে লেজিটিমেট (legitimate) টাওয়ার সিগন্যাল দুর্বল হয়ে পড়েছে। রেড টিম একটি দ্বিমুখী আক্রমণ (two-pronged assault) ডিপ্লয় করে: একটি IMSI Catcher ব্যবহার করে সেলুলার-লেয়ার অ্যাটাক (cellular-layer attack) এবং একটি Evil Twin অ্যাক্সেস পয়েন্ট ব্যবহার করে Wi-Fi-লেয়ার অ্যাটাক (Wi-Fi-layer attack)।

চলুন উভয় কিল চেইন (kill chains) একযোগে বিশ্লেষণ করে দেখি।

Red Team: প্রং ১ — দ্য Stingray (Cellular Layer)

রেড টিম কনফারেন্স রেজিস্ট্রেশন ডেস্কের কাছে একটি ব্যাকপ্যাকের ভেতরে পোর্টেবল IMSI Catcher (Stingray) ডিপ্লয় করে। Stingray একটি শক্তিশালী সেলুলার সিগন্যাল ব্রডকাস্ট করে, যা ভেন্যুতে সার্ভিস দেওয়া লেজিটিমেট টাওয়ারের চেয়ে কৃত্রিমভাবে অনেক বেশি শক্তিশালী।

Stingray লোকাল 4G এবং 5G ফ্রিকোয়েন্সি জ্যাম (jam) করতে শুরু করে, যা আধুনিক সেলুলার প্রোটোকলগুলোর জন্য একটি ডেড জোন (dead zone) তৈরি করে। একই সাথে, এটি একটি 2G সিগন্যাল ব্রডকাস্ট করে। সাধারণ কমার্শিয়াল ডিভাইসের ক্ষেত্রে, যখন baseband তার 4G কানেকশন হারিয়ে ফেলে তখন সেটি প্যানিক করে এবং স্বয়ংক্রিয়ভাবে নিকটস্থ 2G নেটওয়ার্কে ডাউনগ্রেড (downgrade) হয়ে যায়। Stingray তখন সেগুলোকে ক্যাচ (catch) করে। এটি IMSI নাম্বার লগ করা, আনএনক্রিপ্টেড (unencrypted) SMS মেসেজ ইন্টারসেপ্ট করা এবং ভয়েস কল রেকর্ড করতে শুরু করে।

Stingray এক্সিকিউটিভের ডিভাইসটির জন্য স্ক্যান চালাতে থাকে।

Blue Team: ডিফেন্স — LTE-Only Mode

এক্সিকিউটিভের GrapheneOS ডিভাইসটি **LTE-Only mode**-এ (সেকশন ২.১৩) কনফিগার করা। Baseband প্রসেসরকে স্পষ্টভাবে কোনো লিগ্যাসি (legacy) 2G বা 3G প্রোটোকল কোড লোড বা এক্সিকিউট করতে নিষেধ করা হয়েছে।

যখন Stingray 4G সিগন্যাল জ্যাম করে, তখন এক্সিকিউটিভের ডিভাইসটি ডাউনগ্রেড হয় না। এটি সরাসরি রিফিউজ করে। Baseband একটি অ্যাভেইলেবল LTE কানেকশন খুঁজতে থাকে, কিন্তু জ্যামিংয়ের কারণে আশেপাশে কিছুই না পেয়ে "No Service" রিপোর্ট করে।

এক্সিকিউটিভ সিগন্যাল লস হওয়ার এই সাময়িক ঘটনাটি খেয়াল করেন—যা জনবহুল একটি কনফারেন্সে অতি সামান্য একটি ইনকনভিনিয়েন্স। কয়েক মিনিটের মধ্যেই, যখন তারা Stingray-এর রেডিয়াস (radius) থেকে দূরে সরে যান, ডিভাইসটি পুনরায় একটি লেজিটিমেট 4G টাওয়ারের সাথে রিকানেক্ট করে।

Stingray সাধারণ কমার্শিয়াল ডিভাইস ব্যবহারকারী শত শত অংশগ্রহণকারীর IMSI নাম্বার লগ করেছে এবং তাদের SMS মেসেজ ইন্টারসেপ্ট করেছে। কিন্তু এক্সিকিউটিভের GrapheneOS ডিভাইসটি কখনোই Stingray-এর ক্যাপচার লগে (capture log) ধরা পড়েনি। এটি কখনোই কানেক্ট করেনি। এটি কখনোই ট্রান্সমিট করেনি। রেড টিমের ইকুইপমেন্টের কাছে, সেলুলার নেটওয়ার্কে এক্সিকিউটিভের কোনো অস্তিত্বই নেই।

Cellular kill chain: ব্রোকেন (Broken)।

Red Team: প্রং ২ — দ্য Evil Twin (Wi-Fi Layer)

সেলুলার অ্যাটাকটি ব্যর্থ হয়েছে। রেড টিম তাদের সেকেন্ডারি ভেক্টর (secondary vector) অ্যাক্টিভেট করে।

তারা একটি **Evil Twin** অ্যাক্সেস পয়েন্ট ডিপ্লয় করে—একটি রোগ (rogue) Wi-Fi রাউটার যা কনভেনশন সেন্টারের লেজিটিমেট গেস্ট Wi-Fi-এর হুবহু একই নেটওয়ার্ক নাম (SSID) দিয়ে কনফিগার করা: `ConferenceCenter_Guest_WiFi`। Evil Twin লেজিটিমেট রাউটারের চেয়ে অনেক শক্তিশালী সিগন্যাল ব্রডকাস্ট করে, যাতে আশেপাশের ডিভাইসগুলো স্বয়ংক্রিয়ভাবে এর সাথে কানেক্ট হয়ে যায়।

ভিকটিমের ডিভাইস কানেক্ট হওয়ার সাথে সাথেই, তাদের সমস্ত ইন্টারনেট ট্রাফিক রেড টিমের হার্ডওয়্যার দিয়ে রাউট (route) করা হয়। রেড টিম তিনটি সাব-অ্যাটাক (sub-attacks) ডিপ্লয় করে:

Sub-Attack A — DNS Spoofing: যখন একটি কানেক্টেড ডিভাইস `mail.protonmail.com` ভিজিট করার চেষ্টা করে, রেড টিমের DNS সার্ভার কোয়েরিটি (query) ইন্টারসেপ্ট করে এবং একটি পিক্সেল-পারফেক্ট (pixel-perfect) ফিশিং ক্লোনের IP অ্যাড্রেস দিয়ে রেসপন্স করে। ভিকটিম তার ক্রেডেনশিয়াল ফেক সাইটটিতে (fake site) এন্টার করে।

Sub-Attack B — SSL Stripping: রেড টিমের প্রক্সি (proxy) HTTPS কানেকশনকে আনএনক্রিপ্টেড HTTP-তে ডাউনগ্রেড করার চেষ্টা করে, যা তাদেরকে প্লেইন টেক্সট (plain text)-এ ট্রাফিক কনটেন্ট পড়ার সুযোগ করে দেয়।

Sub-Attack C — Traffic Analysis: এমনকি কনটেন্ট যদি এনক্রিপ্টেডও থাকে, রেড টিম এক্সিকিউটিভের অনলাইন অ্যাক্টিভিটি ম্যাপ করতে এবং বিজনেস পার্টনারদের আইডেন্টিফাই করার জন্য ডেস্টিনেশন (destination) IP অ্যাড্রেস এবং ডোমেইন নেমগুলো (domain names) লগ করে।

Blue Team: ডিফেন্স লেয়ার ১ — MAC Anonymity

এক্সিকিউটিভের ডিভাইস Evil Twin-এর সাথে কানেক্ট করার আগেই, প্রথম ডিফেন্সটি ফায়ার (fire) হয়ে গেছে।

GrapheneOS-এর **Per-Connection MAC Randomization** (সেকশন ২.১০) এই স্পেসিফিক Wi-Fi সেশনের জন্য সম্পূর্ণ নতুন এবং র‍্যান্ডম একটি MAC অ্যাড্রেস জেনারেট করে। রেড টিমের অ্যাক্সেস পয়েন্ট কানেকশনটি লগ করে ঠিকই, কিন্তু MAC অ্যাড্রেসটি একটি ডিসপোজেবল ফ্যান্টম (disposable phantom)। এটি এক্সিকিউটিভের পূর্ববর্তী কোনো নেটওয়ার্ক কানেকশনের সাথে কো-রিলেট (correlate) করা যায় না। যদি এক্সিকিউটিভ ডিসকানেক্ট করেন এবং পাঁচ মিনিট পর রিকানেক্ট করেন, তবে MAC অ্যাড্রেসটি পুনরায় পরিবর্তন হয়ে যায়। রেড টিম কোনো পারসিস্টেন্ট (persistent) ট্র্যাকিং প্রোফাইল তৈরি করতে পারে পলতে পারে না।

Blue Team: ডিফেন্স লেয়ার ২ — The VPN Kill Switch

এক্সিকিউটিভের ডিভাইসটি Evil Twin-এর নেটওয়ার্কের সাথে কানেক্ট করে। স্পুফিং অ্যাটাক (spoofing attack) ডিপ্লয় করার জন্য রেড টিমের রাউটার প্রথম DNS কোয়েরিটির জন্য অধীর আগ্রহে অপেক্ষা করতে থাকে।

কোয়েরিটি কখনোই আসে না।

এক্সিকিউটিভের ডিভাইসটি **Always-On** মোডে WireGuard VPN এবং **Block connections without VPN** Kill Switch এনগেজড (engaged) অবস্থায় কনফিগার করা (সেকশন ২.১১)। Wi-Fi কানেকশন এস্টাব্লিশ (establish) হওয়ার সাথে সাথেই, GrapheneOS নেটওয়ার্কিং স্ট্যাক (networking stack) লোকাল নেটওয়ার্কের মাধ্যমে কোনো ট্রাফিক রাউট করতে রিফিউজ করে দেয়। এর পরিবর্তে, এটি অবিলম্বে রিমোট VPN সার্ভারের সাথে WireGuard ক্রিপ্টোগ্রাফিক হ্যান্ডশেক (cryptographic handshake) ইনিশিয়েট (initiate) করে।

কয়েক মিলিসেকেন্ডের মধ্যেই একটি এনক্রিপ্টেড টানেল (encrypted tunnel) এস্টাব্লিশ হয়ে যায়। এই মুহূর্ত থেকে, এক্সিকিউটিভের ডিভাইস থেকে বের হওয়া প্রতিটি প্যাকেট (packet) WireGuard-এর Curve25519 এনক্রিপশনে মোড়ানো থাকে। রেড টিমের রাউটার শুধুমাত্র একটি সিঙ্গেল IP অ্যাড্রেসের (VPN সার্ভার) দিকে প্রবাহিত এনক্রিপ্টেড অর্থহীন ডেটার (gibberish) নিরবচ্ছিন্ন স্ট্রিম দেখতে পায়।

তারা কনটেন্টটি পড়তে পারে না। তারা ডেস্টিনেশন ওয়েবসাইটগুলো আইডেন্টিফাই করতে পারে না। তারা স্পুফড (spoofed) DNS রেসপন্স ইনজেক্ট (inject) করতে পারে না। SSL stripping প্রক্সির কাছে স্ট্রিপ (strip) করার মতো কিছুই থাকে না, কারণ এটি কখনোই TLS হ্যান্ডশেক দেখতে পায় না—এটি WireGuard টানেলের ভেতরে কবরস্থ (buried) থাকে।

Sub-Attacks A, B, and C: সবগুলো ডিনাইড (denied)।

Blue Team: ডিফেন্স লেয়ার ৩ — Private DNS (The Backup Bulkhead)

অ্যাবসোলিউট রিগোর (absolute rigor) নিশ্চিত করার স্বার্থে, চলুন ধরে নিই যে ইনিশিয়াল কানেকশনের সময় WireGuard টানেলটি সাময়িকভাবে ফেইল করেছে (হতে পারে VPN সার্ভার সাময়িকভাবে আনরিচেবল)।

একটি সাধারণ ডিভাইসে, Kill Switch সমস্ত ট্রাফিক ব্লক করে দিত। কিন্তু যদি Kill Switch-এ একটি মাইক্রো-সেকেন্ডের রেস কন্ডিশন (race condition) থাকে, এবং ব্লক এনগেজ হওয়ার আগেই একটি সিঙ্গেল DNS কোয়েরি লোকাল নেটওয়ার্কে লিক (leak) হয়ে যায়, তাহলে কী হবে?

এক্সিকিউটিভের ডিভাইসটিতে `dns.quad9.net` দিয়ে **Private DNS** কনফিগার করা আছে (সেকশন ২.১২)। এমনকি যদি কোনো DNS কোয়েরি VPN টানেল থেকে এস্কেপ (escape) করেও, সেটি DNS-over-TLS এর মাধ্যমে এনক্রিপ্টেড থাকে। রেড টিমের DNS স্পুফিং সার্ভার একটি এনক্রিপ্টেড ব্লব (encrypted blob) রিসিভ করে যা সে পড়তে পারে না, মডিফাই করতে পারে না এবং স্পুফ করতে পারে না। কোয়েরিটি Quad9-এর সার্ভার দ্বারা সুরক্ষিতভাবে রিজলভ (resolve) হয়, এবং রেসপন্সটি এনক্রিপ্টেড TLS চ্যানেলের মাধ্যমে ফিরে আসে।

ব্যাকআপ বাল্কহেড (backup bulkhead) অটুট থাকে।

আফটার-অ্যাকশন রিপোর্ট (After-Action Report)

রেড টিম একটি পাবলিক ভেন্যুতে এক্সিকিউটিভের বিরুদ্ধে একটি সমন্বিত, দ্বিমুখী রেডিও ফ্রিকোয়েন্সি অ্যাসাল্ট (radio frequency assault) ডিপ্লয় করেছিল। উভয় ভেক্টরই পুরোপুরি নিউট্রালাইজ (neutralize) হয়ে গেছে।

আফটার-অ্যাকশন সামারি (After-Action Summary):

- **Stingray (2G Downgrade):** 4G জ্যাম করেছিল এবং ফেক (fake) 2G ব্রডকাস্ট করেছিল। LTE-Only mode দ্বারা প্রতিহত (baseband 2G রিফিউজ করেছে)। **DENIED.**
- **Evil Twin (DNS Spoofing):** DNS কোয়েরি ইন্টারসেপ্ট করেছিল। WireGuard VPN টানেল দ্বারা প্রতিহত (সমস্ত ট্রাফিক এনক্রিপ্টেড)। **DENIED.**
- **Evil Twin (SSL Stripping):** HTTPS ডাউনগ্রেড করার চেষ্টা করেছিল। ট্রাফিক কখনোই ভিজিবল (visible) ছিল না (VPN টানেলের ভেতরে)। **DENIED.**
- **Evil Twin (Traffic Analysis):** ডেস্টিনেশন IP/ডোমেইন লগ করেছিল। VPN সার্ভার IP-র দিকে সিঙ্গেল এনক্রিপ্টেড স্ট্রিম। **DENIED.**

- **Persistent MAC Tracking:** সেশন জুড়ে MAC কো-রিলেট করেছিল। Per-Connection MAC Randomization দ্বারা প্রতিহত। **DENIED.**
- **DNS Leak (Edge Case):** লিক হওয়া প্লেইনটেক্সট DNS ক্যাপচার করেছিল। Private DNS (Quad9-এর প্রতি DNS-over-TLS) দ্বারা প্রতিহত। **DENIED.**

এক্সিকিউটিভ তার তিন দিনের কনফারেন্স সম্পন্ন করেছেন, পাবলিক Wi-Fi-এর সাথে কানেক্ট করেছেন এবং সেলুলার কল করেছেন—সবকিছুই করা হয়েছে কোনো সিঙ্গেল বাইটের (byte) ইন্টেলিজেন্স ইন্টারসেপ্ট হওয়া ছাড়াই। রেড টিমের ইকুইপমেন্ট অন্যান্য শত শত অংশগ্রহণকারীর ক্রেডেনশিয়াল এবং ব্রাউজিং হিস্ট্রি ক্যাপচার করেছে। এক্সিকিউটিভের ডিভাইস এনক্রিপ্টেড নয়েজ (encrypted noise) এবং ডিসপোজেবল ফ্যান্টম আইডেন্টিফায়ার (disposable phantom identifiers) ছাড়া আর কিছুই প্রডিউস করেনি।

দৃশ্যকল্প ৪ (Scenario 4): **Blue Team-এর বিজয়।**

৩.৮ দৃশ্যকল্প ৫ (Scenario 5): The Supply Chain Attack

Threat Classification: Remote / Infrastructure Compromise

Red Team Assets: কম্প্রোমাইজড (compromised) CDN নোড, ফোর্জড (forged) OTA আপডেট প্যাকেজ, নেটওয়ার্ক ইন্টারসেপশন ক্যাপাবিলিটি

Blue Team Assets: GrapheneOS Fortress Blueprint (Android Verified Boot 2.0, হার্ডওয়্যার-ফিউজড (hardware-fused) পাবলিক কী, Titan M2 রোলব্যাক প্রোটেকশন (rollback protection), পিনড (pinned) TLS সার্টিফিকেট)

Objective (Red Team): টার্গেট ডিভাইসে একটি ট্রোজানড (trojaned) অপারেটিং সিস্টেম আপডেট ডেলিভার ও ইন্সটল করা

Objective (Blue Team): সমস্ত আনঅথোরাইজড (unauthorized) ফার্মওয়্যার রিজেক্ট করা। চেইন-অফ-ট্রাস্টের (chain-of-trust) অ্যাবসোলিউট ইন্টেগ্রিটি (absolute integrity) বজায় রাখা।

অপারেশনাল কনটেক্সট (Operational Context)

পূর্ববর্তী প্রতিটি দৃশ্যকল্পে, রেড টিম (Red Team) অপারেটরের ডিভাইসে সরাসরি আক্রমণ করেছে—RAM, রেডিও ফ্রিকোয়েন্সি, বা অ্যাপ্লিকেশন লেয়ারকে টার্গেট করে। এই দৃশ্যকল্পে, রেড টিম এন্ডপয়েন্টটিকে (endpoint) সম্পূর্ণভাবে পরিত্যাগ করে সরাসরি ইনফ্রাস্ট্রাকচার বা কাঠামোর ওপর আক্রমণ চালায়।

সাপ্লাই চেইন অ্যাটাক (supply chain attacks) হলো আধুনিক সাইবার ওয়ারফেয়ারের সবচেয়ে বিধ্বংসী এবং ভয়ংকর অপারেশনগুলোর মধ্যে অন্যতম। কোনো সিঙ্গেল টার্গেটকে হ্যাক করার পরিবর্তে, অ্যাডভারসারি (adversary) ডিস্ট্রিবিউশন পাইপলাইনটিকে কম্প্রোমাইজ করে, এবং আপডেটটিকে পয়জনিং (poisoning) করে যা একই সাথে হাজার হাজার বা লক্ষ লক্ষ ডিভাইসে পুশ করা হয়। ২০২০ সালের SolarWinds অ্যাটাক—যেখানে রাশিয়ান ইন্টেলিজেন্স অপারেটিভরা একটি লেজিটিমেট এন্টারপ্রাইজ আপডেটের মধ্যে ম্যালওয়্যার (malware) ইনজেক্ট করার জন্য সফটওয়্যার বিল্ড সিস্টেমটিকে কম্প্রোমাইজ করেছিল, যা ইউএস ট্রেজারি (United States Treasury) সহ ১৮,০০০ প্রতিষ্ঠানকে ক্ষতিগ্রস্ত করেছিল—এই ভেক্টরের (vector) মারাত্মক ও ধ্বংসাত্মক ক্ষমতা প্রদর্শন করেছে।

এই দৃশ্যকল্পে, একটি রাষ্ট্র-পৃষ্ঠপোষক রেড টিম GrapheneOS-এর আপডেট ডেলিভারি ইনফ্রাস্ট্রাকচারকে টার্গেট করে। তাদের উদ্দেশ্য হলো আপডেট পাইপলাইনে একটি ট্রোজানড অপারেটিং সিস্টেম ইমেজ (trojaned operating system image) ইনজেক্ট করা, যা আপডেট গ্রহণকারী প্রতিটি GrapheneOS ডিভাইসে পারসিস্টেন্ট স্পাইওয়্যার (persistent spyware) ডেলিভার করবে।

Red Team: ফেজ ১ — দ্য ডেলিভারি পাইপলাইন কম্প্রোমাইজ (Compromising the Delivery Pipeline)

রেড টিম আইডেন্টিফাই করে যে GrapheneOS একটি Content Delivery Network (CDN)-এর মাধ্যমে over-the-air (OTA) আপডেট ডিস্ট্রিবিউট করে। তারা একটি CDN এজ নোডের (edge node) বিরুদ্ধে একটি অত্যাধুনিক ইনট্রুশন (intrusion) চালায়—যা নির্দিষ্ট অঞ্চলের ডিভাইসগুলোতে আপডেট ফাইলগুলো ক্যাশ (cache) এবং ডেলিভার করার জন্য দায়িত্বপ্রাপ্ত একটি ভৌগোলিকভাবে ডিস্ট্রিবিউটেড সার্ভার।

রেড টিম CDN নোডে রাইট অ্যাক্সেস (write access) পেয়ে যায়। তারা লেজিটিমেট GrapheneOS OTA আপডেট ফাইলটিকে একটি ট্রোজানড ভার্সন দিয়ে রিপ্লেস করে। ট্রোজানড ইমেজটি আসল আপডেটের একটি পিক্সেল-পারফেক্ট (pixel-perfect) ক্লোন, কিন্তু এর ভেতরে অত্যন্ত গভীরভাবে এমবেডেড (embedded) একটি ব্যাকডোর (backdoor) রয়েছে যা রেড টিমকে পারসিস্টেন্ট রুট অ্যাক্সেস (root access) এবং সাইলেন্টলি এনক্রিপ্টেড কমিউনিকেশন এক্সফিলট্রেট (exfiltrate) করার ক্ষমতা প্রদান করে।

অ্যাফেক্টেড রিজিওনের হাজার হাজার GrapheneOS ডিভাইস লেটেস্ট আপডেটের জন্য CDN-এ পোলিং (polling) শুরু করে। কম্প্রোমাইজড নোডটি তখন ট্রোজানড ইমেজটি সার্ভ করে।

টার্গেট এক্সিকিউটিভের ডিভাইসটি ফাইলটি ডাউনলোড করে।

Blue Team: ডিফেন্স লেয়ার ১ — TLS Certificate Pinning

আপডেট ফাইলটি বিশ্লেষণ করার আগে, আমাদের অবশ্যই বিবেচনা করতে হবে যে রেড টিম CDN কম্প্রোমাইজ করার পরিবর্তে একটি Man-in-the-Middle (MitM) অ্যাটাকের মাধ্যমে ডাউনলোড কানেকশনটি ইন্টারসেপ্ট করতে পারত কি না।

GrapheneOS-এর আপডেট ক্লায়েন্ট স্ট্রিক্ট (strict) **TLS certificate pinning** এনফোর্স করে। ক্লায়েন্টটি ডিভাইসের রুট সার্টিফিকেট স্টোরের (root certificate store) যেকোনো সার্টিফিকেট অথরিটিকে (certificate authority) অন্ধভাবে বিশ্বাস করে না। এটি শুধুমাত্র GrapheneOS আপডেট সার্ভারের নির্দিষ্ট TLS সার্টিফিকেট অ্যাকসেপ্ট করার জন্য হার্ডকোডেড (hardcoded)।

যদি কোনো অ্যাডভারসারি একটি ফ্রডুলেন্ট (fraudulent) TLS সার্টিফিকেট ব্যবহার করে কানেকশনটি ইন্টারসেপ্ট করার চেষ্টা করে (এমনকি সেটি যদি কোনো লেজিটিমেট, কম্প্রোমাইজড সার্টিফিকেট অথরিটি দ্বারাও সাইন করা থাকে), আপডেট ক্লায়েন্ট সার্টিফিকেটের মিসম্যাচটি (mismatch) ডিটেক্ট করে এবং সাথে সাথে কানেকশন টার্মিনেট (terminate) করে দেয়। ডাউনলোড কখনোই শুরু হয় হৈ।

এটি নেটওয়ার্ক ইন্টারসেপশন ভেক্টরকে সম্পূর্ণভাবে নির্মূল করে দেয়। রেড টিমের জন্য একমাত্র কার্যকরী পথ হলো ডিরেক্ট সার্ভার বা CDN কম্প্রোমাইজ, যা তারা এই দৃশ্যকল্পে অর্জন করেছে।

Network interception vector: ডিনাইড (Denied)।

Blue Team: ডিফেন্স লেয়ার ২ — Cryptographic Signature Verification

ট্রোজানড আপডেট ফাইলটি ডিভাইসে ডাউনলোড হয়ে গেছে। GrapheneOS আপডেটার এখন ইন্সটলেশন প্রসেস শুরু করে।

সিস্টেম পার্টিশনে (system partition) আপডেটের সিঙ্গেল বাইটও (byte) রাইট হওয়ার আগে, আপডেটার একটি রিগোরাস (rigorous) **cryptographic signature verification** পারফর্ম করে।

প্রতিটি লেজিটিমেট GrapheneOS রিলিজ প্রজেক্টের প্রাইভেট সাইনিং কী (private signing key) দ্বারা সাইন করা থাকে—যে কীটি শুধুমাত্র GrapheneOS Foundation-এর রিলিজ ইনফ্রাস্ট্রাকচারের কাছে সংরক্ষিত থাকে, যা ইন্টারনেট থেকে এয়ার-গ্যাপড (air-gapped) একটি হার্ডওয়্যার সিকিউরিটি মডিউলে (HSM) স্টোর করা থাকে। এর কোরেসপন্ডিং (corresponding) পাবলিক কী ডিভাইসের বুটলোডার (bootloader) এবং আপডেটার অ্যাপ্লিকেশনে এমবেড করা থাকে।

আপডেটার ডাউনলোডকৃত আপডেট ফাইলটি নেয়, এর সম্পূর্ণ কনটেন্টের একটি ক্রিপ্টোগ্রাফিক হ্যাশ (cryptographic hash) কম্পিউট করে, এবং তারপর ভেরিফাই করে যে হ্যাশটি এমবেডেড পাবলিক কী ব্যবহার করে GrapheneOS Foundation-এর প্রাইভেট কী দ্বারা সঠিকভাবে সাইন করা হয়েছে কি না।

রেড টিমের ট্রোজানড ইমেজটি Foundation-এর প্রাইভেট কী দ্বারা সাইন করা ছিল না। তাদের কাছে এই কী-টির দখল নেই। এমনকি CDN-এর ওপর পূর্ণ নিয়ন্ত্রণ থাকা সত্ত্বেও, তারা ক্রিপ্টোগ্রাফিক সিগনেচারটি ফোর্জ করতে পারবে না, কারণ তাদের একটি 256-bit এলিপটিক কার্ভ (elliptic curve) ফ্যাক্টর (factor) করতে হবে—এটি এমন একটি গাণিতিক অপারেশন যার জন্য অবজারভেবল ইউনিভার্সে (observable universe) বিদ্যমান মোট শক্তির চেয়েও বেশি কম্পিউটেশনাল এনার্জি (computational energy) প্রয়োজন হবে।

সিগনেচার ভেরিফিকেশন ফেইল করে। আপডেটার অবিলম্বে ট্রোজানড ফাইলটি রিজেক্ট করে এবং একটি এরর (error) ডিসপ্লে করে। ইন্সটলেশন আর সামনে এগোয় না। ম্যালিশিয়াস ইমেজের সিঙ্গেল বাইটও সিস্টেম পার্টিশনে রাইট করা হয় না।

Trojaned update: রিজেক্টেড (Rejected)।

Blue Team: ডিফেন্স লেয়ার ৩ — Android Verified Boot 2.0 (AVB)

চলুন দৃশ্যকল্পটিকে এর থিওরেটিক্যাল এক্সট্রিমে (theoretical extreme) এসক্যালেট (escalate) করি। ধরা যাক, রেড টিম কল্পনাতীত সবচেয়ে অত্যাধুনিক সাপ্লাই চেইন অ্যাটাকটি এক্সিকিউট করেছে: তারা ফিজিক্যালি GrapheneOS Foundation-এর এয়ার-গ্যাপড বিল্ড ইনফ্রাস্ট্রাকচারে ব্রিচ (breach) করেছে এবং প্রাইভেট সাইনিং কী-টি চুরি করেছে। তারা লেজিটিমেট কী ব্যবহার করে তাদের ট্রোজানড ইমেজ সাইন করে এবং এটি CDN-এর মাধ্যমে পুশ করে।

আপডেটটি সিগনেচার ভেরিফিকেশনে পাস করে যায়। এটি সিস্টেম পার্টিশনে রাইট করা হয়। আপডেট অ্যাপ্লাই (apply) করার জন্য ডিভাইসটি রিবুট (reboot) করে।

এমনকি এই ক্যাটেস্ট্রোফিক (catastrophic) দৃশ্যকল্পেও, ডিভাইসটি একটি চূড়ান্ত, ইমিউটেবল (immutable) লাইন অব ডিফেন্স দ্বারা প্রটেক্টেড থাকে: **Android Verified Boot 2.0 (AVB)**।

যখন আপনি শুরুতে GrapheneOS ইন্সটল করেছিলেন এবং বুটলোডার লক করেছিলেন (সেকশন ১.৪), তখন ডিভাইসের সিকিউর এলিমেন্ট (Titan M2 চিপ) GrapheneOS-এর পাবলিক সাইনিং কী-গুলোকে এর টেম্পার-রেসিস্ট্যান্ট (tamper-resistant) হার্ডওয়্যার রেজিস্টারে এনরোল (enroll) করেছিল। এই কী-গুলো সিলিকনে ফিউজ (fuse) করা। এগুলো সফটওয়্যার দ্বারা মডিফাই করা যায় না। এগুলো কোনো OTA আপডেট দ্বারা ওভাররাইট (overwrite) করা যায় না।

প্রতিবার যখন ডিভাইসটি বুট করে, অপারেটিং সিস্টেম কার্নেল (kernel) লোড হওয়ার আগে, Titan M2 একটি ক্রিপ্টোগ্রাফিক চেইন-অফ-ট্রাস্ট ভেরিফিকেশন এক্সিকিউট করে:

1. Titan M2-এর ইমিউটেবল বুট ROM বুটলোডার ভেরিফাই করে।
2. বুটলোডার `vbmeta` পার্টিশন ভেরিফাই করে (Verified Boot মেটাডেটা)।
3. `vbmeta` মেটাডেটার মধ্যে প্রতিটি সিস্টেম পার্টিশনের (boot, system, vendor, dtbo) ক্রিপ্টোগ্রাফিক হ্যাশ থাকে।
4. বুটলোডার ফিজিক্যাল স্টোরেজে প্রতিটি পার্টিশনের হ্যাশ কম্পিউট করে এবং `vbmeta`-তে থাকা এক্সপেক্টেড হ্যাশের সাথে সেটি কম্পেয়ার করে।

রেড টিমের ট্রোজানড ইমেজটি যদি অপারেটিং সিস্টেমের একটি সিঙ্গেল বাইটও মডিফাই করে—এমনকি কনফিগারেশন ফাইলে একটি সিঙ্গেল ক্যারেক্টারও (character)—তবে কম্পিউটেড হ্যাশটি `vbmeta`-তে থাকা এক্সপেক্টেড হ্যাশের সাথে ম্যাচ করবে না। ভেরিফিকেশন ফেইল করবে।

যখন AVB ভেরিফিকেশন ফেইল করে, Titan M2 সাইলেন্টলি কম্প্রোমাইজড ইমেজটি বুট করে না। এটি বুট প্রসেসটি সম্পূর্ণভাবে হল্ট (halt) করে দেয় এবং একটি সতর্কবার্তা সম্বলিত স্ক্রিন প্রদর্শন করে, যা নির্দেশ করে যে অপারেটিং সিস্টেমটি টেম্পার করা হয়েছে। ডিভাইসটি কম্প্রোমাইজড কার্নেল লোড করতে রিফিউজ করে।

Tampered firmware: বুট ডিনাইড (Boot denied)।

Blue Team: ডিফেন্স লেয়ার ৪ — Anti-Rollback (The Final Wall)

চূড়ান্ত পদক্ষেপ হিসেবে, ধরা যাক রেড টিম একটি রোলব্যাক অ্যাটাক (rollback attack) করার চেষ্টা করছে—বর্তমান ফার্মওয়্যারটিকে একটি পুরোনো, লেজিটিমেটভাবে সাইন করা ভার্সন দিয়ে রিপ্লেস করা, যার মধ্যে এমন কোনো প্যাচড (patched) ভালনারেবিলিটি রয়েছে যা তারা এক্সপ্লয়েট (exploit) করতে পারে।

Titan M2 ইন্টারনালি একটি মনোটোনিক্যালি ইনক্রিজিং (monotonically increasing) **rollback counter** মেইনটেইন করে। প্রতিবার যখন একটি লেজিটিমেট GrapheneOS আপডেট ইন্সটল হয়, এই কাউন্টারটি ইনক্রিমেন্ট (increment) হয়। হার্ডওয়্যার পার্মানেন্টলি এমন যেকোনো ফার্মওয়্যার ইমেজ বুট করতে রিফিউজ করে, যার ভার্সন নাম্বার বর্তমান কাউন্টার ভ্যালুর চেয়ে কম।

পুরোনো, ভালনারেবল (vulnerable) ফার্মওয়্যারটি লেজিটিমেটভাবে সাইন করা ছিল ঠিকই, কিন্তু এর ভার্সন নাম্বারটি কাউন্টারের নিচে। Titan M2 হার্ডওয়্যার লেভেলে একে রিজেক্ট করে দেয়। রোলব্যাক অ্যাটাকটি ব্যর্থ হয়।

আফটার-অ্যাকশন রিপোর্ট (After-Action Report)

সাপ্লাই চেইন অ্যাটাক—মডার্ন থ্রেট ল্যান্ডস্কেপে (threat landscape) সবচেয়ে ডেভাস্টেটিং (devastating) ক্লাসের সাইবার অপারেশন—চারটি ইন্ডিপেন্ডেন্ট ডিফেন্স লেয়ারে কম্প্রিহেন্সিভলি নিউট্রালাইজ (neutralize) করা হয়েছে।

আফটার-অ্যাকশন সামারি (After-Action Summary):

- **Network Interception (আপডেট ডাউনলোডের উপর MitM):** TLS certificate pinning দ্বারা প্রতিহত। **DENIED.**
- **CDN Compromise (ট্রোজানড আপডেট ফাইল সার্ভ করেছিল):** Cryptographic signature verification দ্বারা প্রতিহত। **DENIED.**
- **Signing Key Theft (চুরি করা কী দিয়ে ট্রোজান সাইন করেছিল):** Verified Boot 2.0 (হার্ডওয়্যার হ্যাশ চেইন) দ্বারা প্রতিহত। **DENIED.**
- **Firmware Rollback (পুরোনো ভালনারেবল ভার্সন ইন্সটল করেছিল):** Anti-rollback counter (Titan M2) দ্বারা প্রতিহত। **DENIED.**

ইমিউটেবল হার্ডওয়্যারে নোঙ্গর করা এই গাণিতিক চেইন-অফ-ট্রাস্ট নিশ্চিত করে যে পৃথিবীর কোনো এনটিটি (entity)—সেটা কোনো কম্প্রোমাইজড সার্ভার হোক, চুরি হওয়া কী হোক, বা কোনো নেশন-স্টেট ইন্টেলিজেন্স এজেন্সি (nation-state intelligence agency) হোক—একটি প্রপারলি কনফিগার করা GrapheneOS ডিভাইসে সাইলেন্টলি আনঅথোরাইজড কোড ডেলিভার করতে পারবে না।

দৃশ্যকল্প ৫ (Scenario 5): **Blue Team-এর বিজয়।**

৩.৯ ডিপ ডাইভ: Exec Spawning বনাম Zygote (Exec Spawning vs. Zygote)

পেগাসাস সিমুলেশনে (অধ্যায় ৩.৪), আমরা দেখেছি কীভাবে Control Flow Integrity এবং `hardened_malloc` মেমরি বাউন্ডারি করাপ্ট করা এবং এক্সিকিউশন ফ্লো হাইজ্যাক করা থেকে অ্যাটাকারকে প্রতিহত করে রেড টিমের (Red Team) এক্সপ্লয়েট চেইনকে (exploit chain) ধ্বংস করে দিয়েছিল। তবে, এই সুরক্ষা স্তরগুলোর নিচে একটি মৌলিক এবং প্রায়শই অদৃশ্য ডিফেন্স মেকানিজম (defense mechanism) কাজ করে—এমন একটি মেকানিজম যা মেমরি এক্সপ্লয়টেশনের (memory exploitation) গাণিতিক ল্যান্ডস্কেপকে আমূল পরিবর্তন করে দেয়।

এটি বোঝার জন্য, আমাদের অবশ্যই পরীক্ষা করে দেখতে হবে কীভাবে স্ট্যান্ডার্ড অ্যান্ড্রয়েড অ্যাপ্লিকেশন লঞ্চ করে এবং কেন সেই প্রক্রিয়ায় একটি ভয়াবহ সিকিউরিটি ফ্ল (security flaw) বা নিরাপত্তা ত্রুটি রয়েছে।

স্ট্যান্ডার্ড অ্যান্ড্রয়েড যেভাবে অ্যাপ লঞ্চ করে: জাইগোট মডেল (The Zygote Model)

অ্যাপ্লিকেশন লঞ্চ করা একটি এক্সপেনসিভ অপারেশন (expensive operation)। অপারেটিং সিস্টেমকে অবশ্যই অ্যাপ্লিকেশন রানটাইম (Android Runtime বা ART) লোড করতে হয়, জাভা ভার্চুয়াল মেশিন (Java Virtual Machine) ইনিশিয়ালাইজ করতে হয়, সিস্টেম লাইব্রেরিগুলো লোড করতে হয়, শেয়ার্ড ফ্রেমওয়ার্কগুলোকে মেমরিতে ম্যাপ করতে হয় এবং তারপর অ্যাপ্লিকেশনের কোড এক্সিকিউট করা শুরু করতে হয়। লিমিটেড হার্ডওয়্যারের প্রাথমিক অ্যান্ড্রয়েড ডিভাইসগুলোতে প্রতিটি অ্যাপ লঞ্চের জন্য শুরু থেকে এই ইনিশিয়ালাইজেশন (initialization) সম্পন্ন করা অত্যন্ত ধীর গতির ছিল।

এই পারফরম্যান্স সমস্যার সমাধান করার জন্য, অ্যান্ড্রয়েডের ইঞ্জিনিয়াররা **Zygote** (জাইগোট) নামের একটি প্রসেস তৈরি করেছিলেন।

Zygote হলো একটি বিশেষ, পারসিস্টেন্ট প্রসেস (persistent process) যা ডিভাইস বুট হওয়ার সময় শুরু হয়। ইনিশিয়ালাইজেশনের সময়, Zygote সম্পূর্ণ অ্যান্ড্রয়েড রানটাইম, সমস্ত সিস্টেম লাইব্রেরি এবং সমস্ত শেয়ার্ড ফ্রেমওয়ার্ক কোডকে এর মেমরি স্পেসে প্রি-লোড (pre-load) করে। তারপর এটি আইডল (idle) অবস্থায় বসে অপেক্ষা করতে থাকে।

আপনি যখন কোনো অ্যাপ্লিকেশন আইকনে ট্যাপ করেন, তখন অপারেটিং সিস্টেম শূন্য থেকে অ্যাপের প্রসেস তৈরি করে না। পরিবর্তে, এটি Zygote-এর কাছে একটি কমান্ড পাঠায় এবং Zygote একটি সিস্টেম কল এক্সিকিউট করে যা `fork()` নামে পরিচিত। `fork()` কলটি Zygote প্রসেসের হুবহু একটি বাইট-ফর-বাইট ক্লোন (byte-for-byte clone) তৈরি করে। এই

ক্লোনটি সম্পূর্ণরূপে ইনিশিয়ালাইজড রানটাইম, সমস্ত প্রি-লোডেড লাইব্রেরি এবং সম্পূর্ণ মেমরি লেআউট ইনহেরিট (inherit) করে। তারপর ক্লোন করা প্রসেসটি এই প্রি-বিল্ট ফাউন্ডেশনের (pre-built foundation) উপর নির্দিষ্ট অ্যাপ্লিকেশনের কোড লোড করে।

এর ফলাফল হলো অত্যন্ত দ্রুতগতির অ্যাপ লঞ্চ। সবচেয়ে ভারী ইনিশিয়ালাইজেশনের কাজটি বুট করার সময় একবার করা হয়েছিল এবং পরবর্তীতে প্রতিটি অ্যাপ লঞ্চ শুধুমাত্র একটি লাইটওয়েট কপি অপারেশন (lightweight copy operation)।

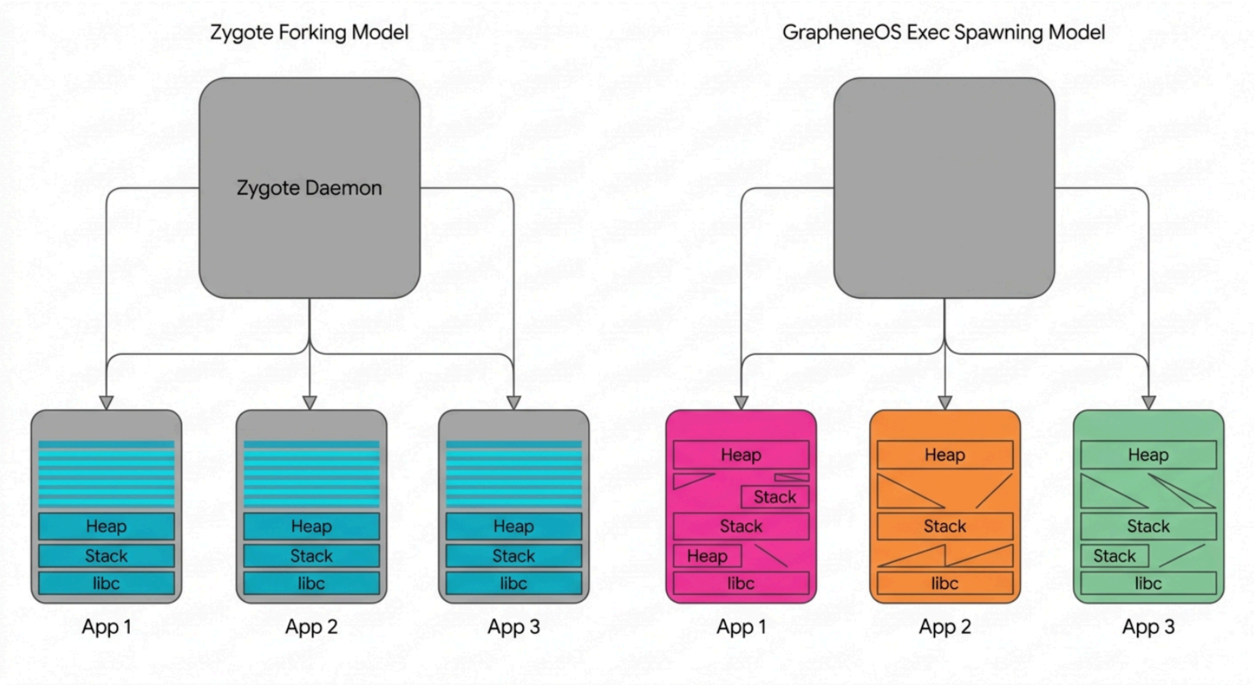
পারফরম্যান্সের দৃষ্টিকোণ থেকে, Zygote চমৎকার। কিন্তু নিরাপত্তার দৃষ্টিকোণ থেকে, Zygote একটি ভয়াবহ বিপর্যয়।

ভয়াবহ ত্রুটি: ASLR ডেস্ট্রাকশন (ASLR Destruction)

আধুনিক অপারেটিং সিস্টেমগুলো **Address Space Layout Randomization (ASLR)** নামক একটি ক্রিটিক্যাল ডিফেন্স মেকানিজমের উপর নির্ভর করে। একটি প্রসেস শুরু হওয়ার পর প্রতিবার সিস্টেম লাইব্রেরি, হিপ (heap) এবং স্ট্যাক (stack) যে মেমরি অ্যাড্রেসগুলোতে লোড হয়, ASLR সেগুলোকে র‍্যান্ডমাইজ (randomize) করে। এর উদ্দেশ্য হলো মেমরিতে নির্দিষ্ট কোড বা ডেটা স্ট্রাকচার ঠিক কোথায় অবস্থিত তা প্রেডিক্ট (predict) বা অনুমান করা অ্যাটাকারের জন্য অসম্ভব করে তোলা।

যদি কোনো অ্যাটাকার ব্রাউজারে একটি ভালনারেবিলিটি (vulnerability) আবিষ্কার করে এবং এক্সিকিউশনটিকে কোনো নির্দিষ্ট সিস্টেম ফাংশনে (যেমন `system()` বা `mprotect()`) রিডাইরেক্ট করতে চায়, তবে তাকে সেই ফাংশনটির সঠিক মেমরি অ্যাড্রেস জানতে হবে। ASLR ব্যতীত, এই অ্যাড্রেসটি সর্বদা একই থাকে এবং অ্যাটাকার সহজেই তার এক্সপ্লয়েটে (exploit) এটিকে হার্ডকোড (hardcode) করে ফেলে। কিন্তু ASLR এর কারণে, প্রতিবার প্রসেস শুরু হওয়ার সময় অ্যাড্রেসটি পরিবর্তিত হয়, ফলে অ্যাটাকারকে অন্ধের মতো অনুমান করতে হয়।

Architectural Topology of Memory Address Spaces



In the standard Zygote model, the preloaded memory template is identically cloned to all child processes, meaning the internal memory addresses of App A are exactly the same as App B. In contrast, Exec Spawning generates an entirely distinct, uniquely randomized memory mapping for every individual application process.

মডার্ন কম্পিউটিংয়ে ASLR হলো অন্যতম একটি মৌলিক এক্সপ্লয়েট মিটিগেশন (exploit mitigation)। কিন্তু Zygote এটিকে সম্পূর্ণভাবে ধ্বংস করে দেয়।

যেহেতু স্ট্যান্ডার্ড অ্যান্ড্রয়েডের প্রতিটি অ্যাপ্লিকেশনই সিঙ্গেল Zygote প্রসেসের একটি `fork()` ক্লোন, তাই **ডিভাইসের প্রতিটি অ্যাপ্লিকেশন হুবহু একই মেমরি লেআউট শেয়ার করে**। আপনার ব্রাউজারে সিস্টেম লাইব্রেরি `libc.so` যে অ্যাড্রেসে লোড হয়, আপনার ক্যামেরা অ্যাপেও সেটি একই অ্যাড্রেসে লোড হয়। আপনার ইমেইল ক্লায়েন্টে রানটাইমের (runtime) ইন্টারনাল স্ট্রাকচারগুলো যে অ্যাড্রেসে থাকে, ফাইল ম্যানেজারেও সেগুলো একই অ্যাড্রেসে থাকে।

ASLR লেআউটটিকে মাত্র একবার র‍্যান্ডমাইজ (randomize) করেছিল—যখন বুট করার সময় Zygote ইনিশিয়ালাইজ (initialize) হয়েছিল। কিন্তু যেহেতু `fork()` একটি আইডেন্টিক্যাল কপি তৈরি করে, তাই ওই একবারের র‍্যান্ডমাইজেশন প্রতিটি অ্যাপ ইনহেরিট (inherit) করে। মূলত, প্রতি বুট সাইকেলে (boot cycle) একবারই র‍্যান্ডমাইজেশন সম্পন্ন হয়, প্রতিবার প্রসেস লঞ্চ হওয়ার সময় নয়।

এর অর্থ হলো, কোনো অ্যাটাকার (attacker) যদি ডিভাইসের *যেকোনো* অ্যাপ্লিকেশন থেকে একটি সিঙ্গেল মেমরি অ্যাড্রেস লিক (leak) করতে পারে (ব্রাউজারে একটি জাভাস্ক্রিপ্ট ইনফরমেশন ডিসক্লোজার ভালনারেবিলিটি ব্যবহার করে এটি করা তুলনামূলকভাবে সহজ), তবে তারা তাৎক্ষণিকভাবে পুরো ডিভাইসের *অন্যান্য সকল অ্যাপ্লিকেশনের* মেমরি লেআউট নিখুঁতভাবে জেনে যায়। ব্রাউজার, ব্যাংকিং অ্যাপ, মেসেঞ্জার—এদের সবাই একই ম্যাপ (map) শেয়ার করে।

অ্যাটাকারকে কোনো কিছু অনুমান করতে হয় না। ব্লুপ্রিন্টটি (blueprint) তাদের কাছেই থাকে।

অ্যাটাকারদের অস্ত্র: ROP চেইন (ROP Chains)

একটি পরিচিত এবং প্রেডিক্টেবল (predictable) মেমরি লেআউট পেয়ে গেলে, অত্যাধুনিক অ্যাটাকাররা **Return-Oriented Programming (ROP)** নামক একটি টেকনিক ডিপ্লয় (deploy) করে।

নিজেদের শেলকোড (shellcode) ইনজেক্ট (inject) করার পরিবর্তে (যা নন-এক্সিকিউটেবল মেমরির মতো ডিফেন্স দ্বারা ব্লক হয়ে যাবে), অ্যাটাকাররা টার্গেটের মেমরিতে আগে থেকে লোড হওয়া লেজিটিমেট কোডের ছোট ছোট ফ্র্যাগমেন্টগুলোকে (fragments) চেইনে (chain) যুক্ত করে। এই ফ্র্যাগমেন্টগুলোকে "গ্যাজেটস" (gadgets) বলা হয়—যেগুলো সাধারণত কয়েকটি ইন্সট্রাকশন দীর্ঘ হয় এবং একটি `return` ইন্সট্রাকশনের মাধ্যমে শেষ হয়।

এই ধরনের ডজন খানেক গ্যাজেট সতর্কতার সাথে একত্রে চেইন (chain) করার মাধ্যমে, অ্যাটাকার শিকারের নিজস্ব লেজিটিমেট বা বৈধ কোড ব্যবহার করেই সম্পূর্ণ একটি ম্যালিসিয়াস প্রোগ্রাম তৈরি করে ফেলে। যেহেতু কোডটি লেজিটিমেট, তাই এটি কোড ইন্টিগ্রিটি চেক (code integrity checks) দ্বারা ফ্ল্যাগড (flagged) বা বাধাগ্রস্ত হয় না। যেহেতু মেমরি লেআউট প্রেডিক্টেবল বা অনুমেয় (Zygote-এর কারণে), তাই অ্যাটাকার তার প্রয়োজনীয় প্রতিটি গ্যাজেটের সঠিক অ্যাড্রেসটি আগে থেকেই জানে।

প্রায় প্রতিটি আধুনিক এবং হাই-টিয়ার এক্সপ্লয়েটের (high-tier exploit) মেরুদণ্ডই হলো ROP চেইন। আর Zygote অ্যাটাকারের হাতে এই রোডম্যাপটি (roadmap) খুব সহজেই তুলে দেয়।

GrapheneOS-এর সমাধান: Exec Spawning (The GrapheneOS Solution: Exec Spawning)

GrapheneOS ফান্ডামেন্টাল বা মৌলিকভাবেই Zygote মডেলটিকে প্রত্যাখ্যান করে।

একটি টেমপ্লেট প্রসেস ক্লোন করার বদলে, GrapheneOS **Exec Spawning** ইমপ্লিমেন্ট (implement) বা বাস্তবায়ন করে। আপনি যখন কোনো অ্যাপ্লিকেশন লঞ্চ করেন, অপারেটিং সিস্টেম তখন `fork()` কল করে না। এর পরিবর্তে, এটি `exec()` কল করে—যা এমন একটি সিস্টেম কল যা সম্পূর্ণ নতুন করে বা স্ক্র্যাচ (scratch) থেকে প্রসেসটিকে তৈরি করে।

Exploit Mitigation Entropy & Isolation Matrix

SECURITY COMPONENT	STANDARD ANDROID (Zygote)	GRAPHENEOS (Exec Spawning)
ASLR Memory Base	**Shared Global** Memory layout identical across apps	**Unique Per-App** Layout randomized cleanly
Stack Canaries	**Shared Global** Canary values inherited from parent	**Unique Per-App** Fresh SSP secrets generated
Scudo Heap Secrets	**Shared Global** Allocator defenses compromised locally	**Unique Per-App** Isolated hardened memory pool
Hardware Memory Tags	**Shared Global** Tagging secrets visible system-wide	**Unique Per-App** Tags securely constrained
Cross-Profile Identifiers	**Leaked via Secrets** Hardware identifier created from state	**Blocked** Side-channel identifier mitigated

Under the standard Zygote model, probabilistic exploit mitigations are generated once at boot and shared globally. This sharing not only defeats ASLR by rendering memory layouts predictable to local attackers but also creates a persistent hardware identifier across isolated user profiles.

Data sources: Synacktiv, USENIX, GrapheneOS, CCERT

Exec Spawning-এর মাধ্যমে যখনই কোনো অ্যাপ্লিকেশন লঞ্চ করা হয়, তখন অপারেটিং সিস্টেম একটি কমপ্লিট এবং ইন্ডিপেন্ডেন্ট ইনিশিয়ালাইজেশন সম্পন্ন করে:

1. একটি ফ্রেশ (fresh) এবং এম্পটি অ্যাড্রেস স্পেস (empty address space) দিয়ে একটি নতুন প্রসেস তৈরি হয়।
2. অ্যান্ড্রয়েড রানটাইম (Android Runtime) ডিস্ক (disk) থেকে একটি **নতুন র‍্যান্ডমাইজড** মেমরি অ্যাড্রেস সেটে লোড হয়।

3. সমস্ত সিস্টেম লাইব্রেরি **নতুন র‍্যান্ডমাইজড** লোকেশনে ম্যাপ করা হয়।
4. হিপ (heap) এবং স্ট্যাক (stack) **নতুন র‍্যান্ডমাইজড** বেস অ্যাড্রেসে (base addresses) ইনিশিয়ালাইজ করা হয়।
5. এই ফ্রেশ এবং ইউনিক ফাউন্ডেশনের ওপর অ্যাপ্লিকেশন কোড লোড করা হয়।

এর ফলে ডিভাইসের প্রতিটি অ্যাপ্লিকেশনের একটি সম্পূর্ণ ইন্ডিপেন্ডেন্ট এবং ইউনিকলি র‍্যান্ডমাইজড (uniquely randomized) মেমরি লেআউট থাকে। আপনার ব্রাউজারের `libc.so` যে অ্যাড্রেসে থাকে, ক্যামেরা অ্যাপের `libc.so`-এর অ্যাড্রেস তার চেয়ে ভিন্ন হয়। আপনার ব্যাংকিং অ্যাপের হিপ এবং মেসেঞ্জারের হিপ সম্পূর্ণ আলাদা লোকেশনে থাকে।

যদি রেড টিম (Red Team) Vanadium ব্রাউজার থেকে কোনো মেমরি অ্যাড্রেস লিক করতে সক্ষম হয়, তবে সেই অ্যাড্রেসটি Signal, ব্যাংকিং অ্যাপ বা অন্য কোনো প্রসেসে অ্যাটাক করার ক্ষেত্রে সম্পূর্ণ অকেজো। প্রতিটি অ্যাপ্লিকেশন ক্রিপ্টোগ্রাফিক দিক থেকে একেকটি ইউনিক ইউনিভার্স (unique universe)। অ্যাটাকারকে অবশ্যই প্রতিটি টার্গেটের মেমরি লেআউট স্বাধীনভাবে আবিষ্কার করতে হবে, যা তাদের এক্সপ্লয়েট চেইনের খরচ এবং জটিলতাকে বহুগুণে বাড়িয়ে দেয়।

ROP চেইন, যা কোড গ্যাজেটগুলোর নিখুঁত অ্যাড্রেস জানার ওপর নির্ভর করে, তা সম্ভাব্যভাবে ইনভিজিবল (infeasible) বা অসম্ভব হয়ে দাঁড়ায়। অ্যাটাকার তখন আর কোনো ম্যাপ দেখে পথ চলতে পারে না; বরং তাকে সম্পূর্ণ অন্ধকারের মধ্যে নেভিগেট (navigate) করতে হয়।

নিরাপত্তার মূল্য (The Cost of Security)

Exec Spawning বিনামূল্যে কাজ করে না। যেহেতু প্রতিটি অ্যাপ্লিকেশন প্রি-লোডেড টেমপ্লেট থেকে ক্লোন না হয়ে স্ক্র্যাচ (scratch) থেকে বা শূন্য থেকে তৈরি হয়, তাই অ্যাপ লঞ্চ হওয়ার সময় সামান্য ধীর হয় (সাধারণত সেকেন্ডের ভগ্নাংশ)। এছাড়া, লাইব্রেরিগুলো copy-on-write পেজের মাধ্যমে শেয়ার করার বদলে ইন্ডিপেন্ডেন্টলি লোড হওয়ার কারণে মেমরি কনজাম্পশনও (memory consumption) কিছুটা বেশি হয়।

GrapheneOS একে একটি অত্যন্ত নগণ্য খরচ হিসেবে বিবেচনা করে। পর্যাপ্ত র‍্যাম (RAM) এবং হাই-স্পিড স্টোরেজ সম্বলিত আধুনিক Pixel হার্ডওয়্যারে এই সামান্য পারফরম্যান্স ওভারহেড কার্যত টের পাওয়া যায় না। এর নিরাপত্তা সুবিধা—পুরো অপারেটিং সিস্টেম জুড়ে ফুল এবং পার-প্রসেস (per-process) ASLR রিস্টোর করা—পরিমাপনাতীত।

এটি পুরো GrapheneOS আর্কিটেকচারে (architecture) বারবার ফিরে আসা একটি থিম। যেখানে স্ট্যান্ডার্ড অ্যান্ড্রয়েড স্পিডের জন্য অপ্টিমাইজ (optimize) করে এবং নিরাপত্তাকে বলি দেয়, সেখানে GrapheneOS নিরাপত্তার জন্য অপ্টিমাইজ করে এবং নূন্যতম ফ্রিকশন (friction) বা বাধা মেনে নেয়। কোনো অ্যাপ লঞ্চ হওয়ার জন্য আপনি সেকেন্ডের যে

ভগ্নাংশ সময় অপেক্ষা করেন, ঠিক সেই সময়েই অপারেটিং সিস্টেম নির্দিষ্ট প্রসেসটির জন্য একটি ইউনিক, র‍্যান্ডমাইজড এবং এক্সপ্লয়েট-রেসিস্ট্যান্ট (exploit-resistant) মেমরি ফোর্ট্রেস (memory fortress) বা দুর্গ তৈরি করে।

স্তরভিত্তিক প্রভাব (The Layered Effect)

Exec Spawning একাকী কাজ করে না। আমাদের আলোচনা করা প্রতিটি মেমরি ডিফেন্সের সাথে এটি একসাথে মিলে কাজ করে:

- `hardened_malloc` হিপ লেআউট র‍্যান্ডমাইজ করে এবং প্রতিটি প্রসেসের মধ্যে গার্ড পেজ (guard pages) ইনসার্ট করে।
- ARM MTE প্রতিটি মেমরি অ্যালোকেশনকে একটি হার্ডওয়্যার-এনফোর্সড কালার (hardware-enforced color) দিয়ে ট্যাগ করে।
- Control Flow Integrity একটি কম্পাইলার-জেনারেটেড (compiler-generated) হোয়াইটলিস্টের (whitelist) বিপরীতে প্রতিটি ফাংশন কলের টার্গেট ভ্যালিডেট (validate) করে।
- Exec Spawning নিশ্চিত করে যে সম্পূর্ণ ফাউন্ডেশনটি—প্রতিটি লাইব্রেরি, স্ট্যাক এবং হিপের বেস অ্যাড্রেসগুলো—প্রতিটি প্রসেসের জন্য ইন্ডিপেন্ডেন্টলি র‍্যান্ডমাইজড।

একত্রিতভাবে, এই চারটি স্তর এমন একটি মেমরি সিকিউরিটি পসচার (memory security posture) তৈরি করে যেখানে অ্যাটাকারকে একই সাথে র‍্যান্ডমাইজড বেস অ্যাড্রেস, র‍্যান্ডমাইজড হিপ লেআউট, হার্ডওয়্যার-ট্যাগড (hardware-tagged) মেমরি বাউন্ডারি এবং কম্পাইলার-এনফোর্সড এক্সিকিউশন ফ্লো ভ্যালিডেশন (compiler-enforced execution flow validation) বাইপাস করতে হয়। আর এ সবকিছুই করতে হয় একটি সিঙ্গেল প্রসেসের বিরুদ্ধে একটি সিঙ্গেল এক্সপ্লয়েট চেইনের মাধ্যমে, এটি জেনেই যে তার অর্জিত এই তথ্য ডিভাইসের অন্য কোনো প্রসেসের বিরুদ্ধে সম্পূর্ণ অকেজো।

এই গাণিতিক হিসাবটি ব্যাপকভাবে এবং চূড়ান্তভাবে ব্লু টিমের (Blue Team) অনুকূলে কাজ করে।

৩.১০ সিনারিও ৬: দ্য ম্যালিশিয়াস অ্যাপ এস্কেপ (Scenario 6: The Malicious App Escape)

থ্রেট ক্লাসিফিকেশন (Threat Classification): লোকাল অ্যাপ্লিকেশন / স্যান্ডবক্স এস্কেপ অ্যাটেম্পট (Local Application / Sandbox Escape Attempt)
রেড টিম অ্যাসেটস (Red Team Assets): ট্রোজানাইজড এপিকে (Trojanized APK) (একটি ইউটিলিটি অ্যাপের ছদ্মবেশে), এমবেডেড প্রিভিলেজ এসক্যালেশন এক্সপ্লয়েট (embedded privilege escalation exploit), ক্লিপবোর্ড হার্ভেস্টার মডিউল (clipboard harvester module), ডেটা এক্সফিলট্রেশন পেলোড (data exfiltration payload)
ব্লু টিম অ্যাসেটস (Blue Team Assets): GrapheneOS ফোর্ট্রেস ব্লুপ্রিন্ট (হার্ডেনড SELinux, Scoped Storage, নেটওয়ার্ক পারমিশন টগল, ক্লিপবোর্ড অ্যাক্সেস রেস্ট্রিকশন, exec spawning, কোনো রুট বাইনারি নেই)
উদ্দেশ্য (Red Team): অ্যাপ্লিকেশন স্যান্ডবক্স এস্কেপ (sandbox escape) করা, অন্যান্য অ্যাপ থেকে ক্রেডেনশিয়াল (credentials) চুরি করা এবং একটি রিমোট সার্ভারে ডেটা এক্সফিলট্রেট বা পাচার করা
উদ্দেশ্য (Blue Team): ম্যালিশিয়াস অ্যাপটিকে এর নিজস্ব স্যান্ডবক্সের মধ্যে আটকে রাখা। সকল ল্যাটারাল মুভমেন্ট (lateral movement) বা পার্শ্ববর্তী অ্যাক্সেস প্রতিহত করা।

অপারেশনাল কন্টেক্সট (Operational Context)

সব অ্যাটাক বা আক্রমণ মিলিয়ন-ডলার মূল্যের জিরো-ক্লিক এক্সপ্লয়েট চেইন (zero-click exploit chains) অথবা অত্যাধুনিক রেডিও ফ্রিকোয়েন্সি ইন্টারসেপশনের (radio frequency interception) মাধ্যমে আসে না। রিয়েল-ওয়ার্ল্ড বা বাস্তব জগতের মোবাইল কম্প্রোমাইজগুলোর (mobile compromises) সিংহভাগই ঘটে সবচেয়ে সাধারণ ভেক্টরের (vector) মাধ্যমে: ব্যবহারকারী নিজেই একটি ম্যালিশিয়াস অ্যাপ্লিকেশন ইনস্টল করে বসেন।

এই সিনারিওতে, টার্গেট—যিনি একজন অ্যাক্টিভিস্ট—টেলিগ্রাম (Telegram) গ্রুপে শেয়ার করা একটি থার্ড-পার্টি APK রিপোজিটরি থেকে জনপ্রিয় একটি ফ্ল্যাশলাইট (flashlight) ইউটিলিটি ডাউনলোড করেন। অ্যাপ্লিকেশনটি ঠিক এর বিজ্ঞাপনের মতোই কাজ করে। ফ্ল্যাশলাইটটি অন এবং অফ হয়। কিন্তু, এই লেজিটিমেট বা বৈধ ইউজার ইন্টারফেসের (UI) আড়ালে লুকিয়ে আছে একটি অত্যাধুনিক ট্রোজান পেলোড (trojan payload), যাকে চারটি ভিন্ন ভিন্ন অ্যাটাক ফেজ (attack phases) বা ধাপে কাজ করার জন্য ইঞ্জিনিয়ারিং করা হয়েছে।

অ্যাক্টিভিস্ট অ্যাপটিকে তার কোয়ারেন্টাইনড (quarantined) "Daily" প্রোফাইলে ইনস্টল করেছেন, যেখানে Sandboxed Google Play নেই। তিনি অ্যাপটিকে শূন্য (zero) অপশনাল পারমিশন দিয়েছেন। চলুন দেখা যাক কী ঘটে।

রেড টিম: ফেজ ১ — ক্লিপবোর্ড হারভেস্টিং (Red Team: Phase 1 — Clipboard Harvesting)

ট্রোজানটির প্রথম উদ্দেশ্য হলো প্যাসিভ ইন্টেলিজেন্স গ্যাদারিং (passive intelligence gathering)। আধুনিক ব্যবহারকারীরা প্রতিনিয়তই তাদের সিস্টেম ক্লিপবোর্ডে সংবেদনশীল ডেটা কপি করেন: পাসওয়ার্ড ম্যানেজার থেকে পাসওয়ার্ড, ক্রিপ্টোকারেন্সি ওয়ালেট অ্যাড্রেস, টু-ফ্যাক্টর অথেনটিকেশন কোড (2FA) এবং ব্যাংকিং ডিটেইলস।

স্ট্যান্ডার্ড অ্যান্ড্রয়েডে (Android 12-এর আগের ভার্সনগুলোতে), ব্যাকগ্রাউন্ডে রান করা যেকোনো অ্যাপ্লিকেশন ব্যবহারকারীর কোনো নোটিফিকেশন বা পারমিশন ছাড়াই নীরবে ক্লিপবোর্ড পড়তে পারত। ম্যালিশিয়াস অ্যাপগুলো এটি অত্যন্ত অ্যাগ্রেসিভভাবে (aggressively) এক্সপ্লয়েট করত, কপি করা ক্রেডেনশিয়ালগুলোর একটি নিরবচ্ছিন্ন স্রোত তারা নীরবে হারভেস্ট (harvest) বা সংগ্রহ করত।

ট্রোজানটি তার ক্লিপবোর্ড হার্ভেস্টার মডিউলটি (clipboard harvester module) অ্যাক্টিভেট বা সক্রিয় করে এবং সিস্টেম ক্লিপবোর্ডকে পোল (poll) করা শুরু করে।

ব্লু টিম: ডিফেন্স — ক্লিপবোর্ড অ্যাক্সেস রেস্ট্রিকশনস (Blue Team: Defense — Clipboard Access Restrictions)

GrapheneOS কঠোর ক্লিপবোর্ড অ্যাক্সেস কন্ট্রোল এনফোর্স (enforce) করে যা স্ট্যান্ডার্ড অ্যান্ড্রয়েডের প্রটেকশনকেও বহুগুণে ছাড়িয়ে যায়।

GrapheneOS-এ, ব্যাকগ্রাউন্ড অ্যাপ্লিকেশনগুলোকে **ক্লিপবোর্ডে অ্যাক্সেস সম্পূর্ণভাবে ডিনাই (completely denied) করা হয়**। অপারেটিং সিস্টেম একটি কঠোর পলিসি প্রয়োগ করে: কেবলমাত্র সেই অ্যাপ্লিকেশনটিই ক্লিপবোর্ডের কন্টেন্ট পড়ার অনুমতি পায় যেটি বর্তমানে ফোরগ্রাউন্ডে (foreground) রয়েছে—অর্থাৎ যেটি স্ক্রিনে অ্যাক্টিভভাবে প্রদর্শিত হচ্ছে এবং ইউজারের ইনপুট রিসিভ করছে।

ট্রোজানাইজড ফ্ল্যাশলাইট অ্যাপটি ব্যাকগ্রাউন্ডে রান করছে (ফ্ল্যাশলাইট অন করার পর ব্যবহারকারী অ্যাপটি ক্লোজ করে দিয়েছেন)। ক্লিপবোর্ড হার্ভেস্টার মডিউলটি ক্লিপবোর্ড রিড করার জন্য সিস্টেম এপিআইকে (system API) কল করে। GrapheneOS কলটিকে ইন্টারসেপ্ট (intercept) করে এবং একটি এম্পটি (empty) বা খালি রেজাল্ট রিটার্ন করে। ট্রোজানটি কিছুই রিসিভ করে না।

এমনকি অ্যাপটি যদি খুব অল্প সময়ের জন্য ফোরগ্রাউন্ডেও আসে, যেকোনো অ্যাপ ক্লিপবোর্ডে অ্যাক্সেস করা মাত্রই GrapheneOS স্ক্রিনের নিচে একটি স্পষ্ট টোস্ট নোটিফিকেশন (toast notification) প্রদর্শন করে ব্যবহারকারীকে সতর্ক করে: *"Flashlight app pasted from your clipboard."* অ্যাপটি কোনোভাবেই এই নোটিফিকেশনটিকে সাপ্রেস

(suppress) বা গোপন করতে পারে না।

ক্লিপবোর্ড হারভেস্টিং: ডিনাইড (Clipboard harvesting: Denied)।

রেড টিম: ফেজ ২ — ল্যাটারাল ডেটা থেফট (স্যান্ডবক্স এস্কেপ) (Red Team: Phase 2 — Lateral Data Theft (Sandbox Escape))

ক্লিপবোর্ড অ্যাটাকটি ব্যর্থ হয়েছে। ট্রোজানটি এবার এসক্যালেট (escalate) করে। এটি এখন সরাসরি ডিভাইসে ইনস্টল করা অন্যান্য অ্যাপ্লিকেশনের ফাইল সিস্টেম এবং ডেটাবেস অ্যাক্সেস করার চেষ্টা করে—বিশেষ করে, এটি কনভারসেশন লগ (conversation logs) চুরি করার জন্য Signal মেসেজিং অ্যাপের এনক্রিপ্টেড ডেটাবেসকে টার্গেট করে।

লিগ্যাসি (legacy) অ্যান্ড্রয়েডে, ব্রড স্টোরেজ পারমিশন (broad storage permissions) থাকা অ্যাপ্লিকেশনগুলো শেয়ার্ড ফাইল সিস্টেমের মধ্য দিয়ে ট্রাভার্স (traverse) করে অন্য অ্যাপের ডেটা অ্যাক্সেস করতে পারত। ট্রোজানটি Signal-এর ইন্টারনাল ডেটা ডিরেক্টরি থেকে ডেটা রিড করার চেষ্টা করে: `/data/data/org.thoughtcrime.securesms/`।

ব্লু টিম: ডিফেন্স — হার্ডেনড SELinux এবং স্কোপড স্টোরেজ (Blue Team: Defense — Hardened SELinux and Scoped Storage)

দুটি স্বাধীন ডিফেন্স লেয়ার বা প্রতিরক্ষামূলক স্তর তাৎক্ষণিকভাবে এই অ্যাটাকটিকে নিষ্ক্রিয় করে দেয়।

Scoped Storage: GrapheneOS অত্যন্ত কঠোরভাবে অ্যান্ড্রয়েডের Scoped Storage আর্কিটেকচার প্রয়োগ করে। প্রতিটি অ্যাপ্লিকেশন তার নিজস্ব প্রাইভেট ডেটা ডিরেক্টরির মধ্যে সীমাবদ্ধ থাকে। ট্রোজানটি কেবলমাত্র `/data/data/com.fake.flashlight/`-এর মধ্যে থাকা নিজের ফাইলগুলোই দেখতে পায়। যখন এটি Signal-এর ডিরেক্টরি অ্যাক্সেস করার চেষ্টা করে, তখন ফাইল সিস্টেম `Permission Denied` রিটার্ন করে। ডিরেক্টরিটি যে শুধু ইনঅ্যাক্সেসিবল (inaccessible) তা-ই নয়; ট্রোজানের দৃষ্টিকোণ থেকে এর কোনো অস্তিত্বই নেই।

Hardened SELinux: GrapheneOS একটি **হার্ডেনড SELinux ম্যান্ডেটরি অ্যাক্সেস কন্ট্রোল পলিসি (hardened SELinux mandatory access control policy)** প্রয়োগ করে যা আপস্ট্রিম (upstream) অ্যান্ড্রয়েডের চেয়ে উল্লেখযোগ্য মাত্রায় বেশি রেস্ট্রিক্টিভ (restrictive) বা কঠোর। ডিভাইসের প্রতিটি প্রসেসকে একটি কঠোর SELinux

সিকিউরিটি কন্টেক্সট (context) বা ডোমেইনে (domain) অ্যাসাইন (assign) করা হয়। ট্রোজানাইজড ফ্ল্যাশলাইট অ্যাপটি `untrusted_app` ডোমেইনে রান করে। অন্যদিকে, Signal-এর ডেটা ডিরেক্টরিটিকে এমন একটি সিকিউরিটি কন্টেক্সট দিয়ে লেবেল করা থাকে যা কেবল `org.thoughtcrime.securesms` ডোমেইন অ্যাক্সেস করতে পারে।

এমনকি ট্রোজানটি যদি কোনো ফাইল সিস্টেম ভালনারেবিলিটি আবিষ্কার করেও ফেলে যা Scoped Storage বাইপাস করতে সক্ষম, তবুও SELinux কার্নেল মডিউল স্বাধীনভাবে ম্যান্ডেটরি অ্যাক্সেস কন্ট্রোল লেয়ারে (MAC) অ্যাক্সেসটিকে ডিনাই বা প্রত্যাখ্যান করবে। SELinux স্ট্যান্ডার্ড পারমিশন মডেলের নিচে অপারেট করে—এটি ডিভাইসের সমস্ত রিসোর্স অ্যাক্সেসের চূড়ান্ত এবং আপসহীন নির্ধারক (arbiter)।

ল্যাটারাল ডেটা থেফট: ডিনাইড (Lateral data theft: Denied)।

রেড টিম: ফেজ ৩ — রুট-এ প্রিভিলেজ এসক্যালেশন (Red Team: Phase 3 — Privilege Escalation to Root)

ট্রোজানটির সর্বোচ্চ মূল্যের বা হাইয়েস্ট-ভ্যালু (highest-value) পেলোডটি হলো একটি প্রিভিলেজ এসক্যালেশন এক্সপ্লয়েট (privilege escalation exploit)। এটি কার্নেল সাবসিস্টেমের (kernel subsystem) একটি পরিচিত ভালনারেবিলিটিকে টার্গেট করে, যার লক্ষ্য হলো এর প্রসেসটিকে সীমাবদ্ধ `untrusted_app` ডোমেইন থেকে উন্নীত করে `root` (superuser) ডোমেইনে নিয়ে যাওয়া। রুট অ্যাক্সেস পেয়ে গেলে, ট্রোজানটি সমস্ত স্যান্ডবক্সিং বাইপাস করতে পারবে, যেকোনো অ্যাপের ডেটা রিড করতে পারবে এবং একটি পারসিস্টেন্ট ব্যাকডোর (persistent backdoor) ইনস্টল করতে সক্ষম হবে।

এক্সপ্লয়েটটি ফায়ার (fires) করে। এটি কার্নেলের একটি ডেটা স্ট্রাকচার করাপ্ট (corrupt) করার জন্য এবং প্রসেসটির সিকিউরিটি ক্রেডেনশিয়ালগুলো ওভাররাইট (overwrite) করার জন্য একটি ভালনারেবল সিস্টেম কল (vulnerable system call) ইনভোক (invoke) করার চেষ্টা করে।

ব্লু টিম: ডিফেন্স — দ্য লেয়ার্ড রুট ডিনায়াল (Blue Team: Defense — The Layered Root Denial)

GrapheneOS প্রিভিলেজ এসক্যালেশনের বিরুদ্ধে একটি চরম বৈরী (hostile) এনভায়রনমেন্ট বা পরিবেশ উপস্থাপন করে:

১. কোনো রুট বাইনারি নেই (No Root Binaries): GrapheneOS-এ `su` (superuser) বাইনারি বা কোনো রুট ম্যানেজমেন্ট ফ্রেমওয়ার্ক অন্তর্ভুক্ত থাকে না। রুটেড (rooted) কাস্টম রমগুলোতে, ম্যালওয়্যারকে এলিভেটেড প্রিভিলেজ (elevated privileges) রিকোয়েস্ট করার জন্য শ্রেফ `su` বাইনারিটিকে কল করলেই চলে। কিন্তু GrapheneOS-এ, এই বাইনারিটির কোনো অস্তিত্বই নেই। ট্রোজানটির এক্সপ্লয়েট বা হাইজ্যাক করার মতো কোনো লেজিটিমেট রুট এসক্যালেশন পাথওয়ে (pathway) এখানে নেই।

২. seccomp-bpf সিস্টেম কল ফিল্টারিং (seccomp-bpf System Call Filtering): এক্সপ্লয়েটটি যে ভালনারেবল সিস্টেম কলটিকে টার্গেট করে, তা GrapheneOS-এর অ্যাগ্রেসিভ seccomp-bpf পলিসি দ্বারা ফিল্টার করা হয়ে থাকে। `untrusted_app` ডোমেইনটিকে শুধুমাত্র অনুমোদিত সিস্টেম কলগুলোর একটি অত্যন্ত সংকীর্ণ হোয়াইটলিস্টের (whitelist) মধ্যে সীমাবদ্ধ রাখা হয়। এক্সপ্লয়েটের টার্গেট সিসকলটি (syscall) সেই হোয়াইটলিস্টে নেই। ফলে কার্নেল এটি এক্সিকিউট করতে অস্বীকার করে এবং একটি এরর বা ত্রুটি রিটার্ন করে।

৩. hardened_malloc এবং MTE: সিসকল ফিল্টারটি বাইপাস করা হলেও, এক্সপ্লয়েটটি কার্নেল ডেটা স্ট্রাকচার করাপ্ট করার যে প্রচেষ্টা চালায়, তাকে অবশ্যই `hardened_malloc`-এর র‍্যান্ডমাইজড লেআউট এবং ARM MTE-এর হার্ডওয়্যার ট্যাগ ভেরিফিকেশন (hardware tag verification) অতিক্রম করতে হবে। করাপ্ট করার এই প্রচেষ্টাটি একটি ট্যাগ মিসম্যাচ (tag mismatch) ট্রিগার করে এবং ক্রেডেনশিয়াল ওভাররাইট সম্পন্ন হওয়ার আগেই সিপিইউ (CPU) প্রসেসটিকে কিল (kill) করে দেয়।

৪. লকড বুটলোডার (Locked Bootloader): তাত্ত্বিকভাবে কোনো অলৌকিক চেইনের (chain of miracles) মাধ্যমে যদি ট্রোজানটি ক্ষণস্থায়ী রুট অ্যাক্সেস অর্জনও করে ফেলে, তবুও এটি কোনো পারসিস্টেন্ট ব্যাকডোর (persistent backdoor) ইনস্টল করতে পারবে না। লক করা বুটলোডার এবং Verified Boot 2.0 এটি নিশ্চিত করে যে, সিস্টেম পার্টিশনের যেকোনো পরিবর্তন পরবর্তী রিবুটেই শনাক্ত হয়ে যাবে এবং টেম্পার (tamper) করা পার্টিশনটিকে সাথে সাথেই রিজেক্ট বা বাতিল করা হবে।

প্রিভিলেজ এসক্যালেশন: ডিনাইড অ্যাট ফোর ইন্ডিপেন্ডেন্ট লেয়ারস (Privilege escalation: Denied at four independent layers)।

রেড টিম: ফেজ ৪ — ডেটা এক্সফিলট্রেশন (Red Team: Phase 4 — Data Exfiltration)

ট্রোজানটি ক্লিপবোর্ড হারভেস্ট করতে, অন্যান্য অ্যাপের ডেটা অ্যাক্সেস করতে বা প্রিভিলেজ এসক্যালেট করতে পুরোপুরি ব্যর্থ হয়েছে। শেষ অবলম্বন হিসেবে, এটি যে সামান্য ডেটা অ্যাক্সেস করতে পারে (যেমন- এর নিজস্ব ইন্টারনাল লগ এবং ডিভাইসের নেটওয়ার্ক কনফিগারেশন) তা একটি রিমোট কমান্ড-অ্যান্ড-কন্ট্রোল সার্ভারে (C2 server) এক্সফিলট্রেট (exfiltrate) বা পাচার করার চেষ্টা করে।

ব্লু টিম: ডিফেন্স — নেটওয়ার্ক পারমিশন টগল (Blue Team: Defense — Network Permission Toggle)

অ্যাক্টিভিস্ট ফোর্ট্রেস ব্লুপ্রিন্ট (Fortress Blueprint) অনুসরণ করেছিলেন। ফ্ল্যাশলাইট অ্যাপটি ইনস্টল করার পরপরই তিনি এর **Network permission** (অধ্যায় ২.৫) রিভোক (revoke) বা বাতিল করে দিয়েছিলেন।

ট্রোজানটি তার রিমোট সার্ভারের সাথে একটি TCP সকেট (TCP socket) ওপেন করার চেষ্টা করে। GrapheneOS কার্নেল লেভেলেই এই কলটিকে ইন্টারসেপ্ট করে। নেটওয়ার্ক ডিনাইড (Network is denied)। ট্রোজানটি DNS-এর মাধ্যমে সার্ভারের ডোমেইন রিজলভ (resolve) করার চেষ্টা করে। ডিনাইড। এটি একটি UDP প্যাকেট (UDP packet) সেন্ড করার চেষ্টা করে। ডিনাইড।

ট্রোজানটির কাছে যে সামান্য ইন্টেলিজেন্স রয়েছে, তা সে ট্রান্সমিট বা প্রেরণ করতে অক্ষম। এটি মূলত সাউন্ডপ্রুফ (soundproof) বা শব্দভেদ্য রুমে আটকে পড়া একজন গুপ্তচর।

ডেটা এক্সফিলট্রেশন: ডিনাইড (Data exfiltration: Denied)।

আফটার-অ্যাকশন রিপোর্ট (After-Action Report)

আফটার-অ্যাকশন সামারি (After-Action Summary):

- **ফেজ ১ — ক্লিপবোর্ড হারভেস্টিং:** ব্যাকগ্রাউন্ড ক্লিপবোর্ড অ্যাক্সেস ডিনায়াল + টোস্ট অ্যালার্ট (toast alert) দ্বারা পরাস্ত। **DENIED.**
- **ফেজ ২ — ল্যাটারাল ফাইল সিস্টেম অ্যাক্সেস:** Scoped Storage + হার্ডেনড SELinux MAC দ্বারা পরাস্ত। **DENIED.**

- **ফেজ ৩ — কার্নেল প্রিভিলেজ এসক্যালেশন:** কোনো `su` বাইনারি না থাকা + seccomp-bpf + hardened_malloc + MTE দ্বারা পরাস্ত। **DENIED.**
- **ফেজ ৪ — C2 সার্ভারে ডেটা এক্সফিলট্রেশন:** নেটওয়ার্ক পারমিশন রিভোকড থাকার কারণে পরাস্ত। **DENIED.**

ম্যালিশিয়াস অ্যাপটি সম্পূর্ণভাবে কন্টেইনড (contained) বা অবরুদ্ধ। এটি তার নিজের খালি স্যান্ডবক্সের বাইরে কিছুই দেখতে পায় না। এটি কারও সাথেই যোগাযোগ করতে পারে না। এটি এসক্যালেট করে কোথাও পৌঁছাতে পারে fix পৌঁছাতে পারে না। ট্রোজানটি মূলত কংক্রিটের এমন একটি জেলের বন্দি, যার কোনো দরজা নেই, কোনো জানালা নেই এবং কোনো রেডিও নেই।

সিনারিও ৬: **ব্লু টিম ভিক্টরি (Blue Team victory)**।

৩.১১ ডিপ ডাইভ: IOMMU এবং হার্ডওয়্যার আইসোলেশন (Deep Dive: IOMMU and Hardware Isolation)

এই বইটির শুরু থেকেই আমরা প্রধানত GrapheneOS-এর সফটওয়্যার ডিফেন্স বা সুরক্ষাব্যবস্থাগুলোর ওপর গভীরভাবে ফোকাস করেছি—মেমরি অ্যালোকেটর, SELinux পলিসি, সিস্টেম কল ফিল্টার এবং ক্রিপ্টোগ্রাফিক ভেরিফিকেশন চেইনগুলো নিয়ে আলোচনা করেছি। কিন্তু পরিশেষে, সফটওয়্যার ডিফেন্সগুলো মূলত ফিজিক্যাল হার্ডওয়্যারেই এক্সিকিউট (execute) হয়। যদি হার্ডওয়্যার নিজেই কম্প্রোমাইজড বা আপসহীন হয়ে পড়ে, তবে কোনো পরিমাণ সফটওয়্যার ইঞ্জিনিয়ারিংই আপনাকে রক্ষা করতে পারবে না।

এই ডিপ ডাইভে (deep dive), আমরা আধুনিক Pixel ডিভাইসগুলোর (এবং আসন্ন Motorola পার্টনারশিপ হার্ডওয়্যারের) ফিজিক্যাল আর্কিটেকচার বা গঠনতন্ত্র পরীক্ষা করে দেখব। এর মাধ্যমে আমরা বুঝতে পারব কীভাবে GrapheneOS সিলিকন-লেভেলের আইসোলেশন (silicon-level isolation) কাজে লাগিয়ে এমনসব থ্রেট বা হুমকিকে কন্টেইন (contain) করে, যেগুলোর উৎপত্তি স্বয়ং ডিভাইসের নিজস্ব সার্কিট্রির (circuitry) ভেতর থেকেই হয়।

দ্য মাল্টি-প্রসেসর রিয়েলিটি (The Multi-Processor Reality)

একটি প্রচলিত ভুল ধারণা হলো, একটি স্মার্টফোনে শুধুমাত্র একটি মাত্র চিপ (chip) থাকে। কিন্তু বাস্তবে, একটি আধুনিক Pixel ডিভাইস হলো একাধিক স্বাধীন এবং স্পেশালাইজড (specialized) প্রসেসরের একটি কনস্টেলেশন (constellation), যার প্রতিটি নিজস্ব ফার্মওয়্যার রান করে:

- **অ্যাপ্লিকেশন প্রসেসর বা Application Processor (AP):** এটি হলো প্রাইমারি CPU (Google Tensor) যা GrapheneOS রান করে, ডিসপ্লে রেন্ডার (render) করে এবং আপনার অ্যাপ্লিকেশনগুলো এক্সিকিউট করে।
- **সেলুলার বেসব্যান্ড প্রসেসর (Cellular Baseband Processor):** এটি একটি ডেডিকেটেড মাইক্রো-কম্পিউটার (সাধারণত Samsung Shannon বা Qualcomm মডেম), যা 2G/3G/4G/5G প্রোটোকলের মাধ্যমে সেলুলার টাওয়ারের সাথে নেগোসিয়েট (negotiate) করার দায়িত্ব পালন করে।
- **Wi-Fi/Bluetooth Controller:** এটি একটি সম্পূর্ণ আলাদা চিপ যা সমস্ত শর্ট-রেঞ্জ বা স্বল্প পাল্লার ওয়্যারলেস কমিউনিকেশন পরিচালনা করে।
- **Titan M2 Secure Element:** এটি একটি ফিজিক্যালি আইসোলেটেড সিকিউরিটি কো-প্রসেসর (security coprocessor), যা এনক্রিপশন কি (encryption keys), Verified Boot এবং অথেনটিকেশন রেট লিমিটিং (authentication rate limiting) ম্যানেজ করে।

- **Image Signal Processor (ISP):** এটি একটি ডেডিকেটেড চিপ যা ক্যামেরা সেন্সর থেকে পাওয়া র (raw) ডেটা প্রসেস করে।
- **GPU:** এটি একটি আলাদা গ্রাফিক্স প্রসেসিং ইউনিট (graphics processing unit), যা রেন্ডারিং (rendering) এবং ডিসপ্লে কম্পোজিটিংয়ের (display compositing) কাজ পরিচালনা করে।

এই প্রসেসরগুলোর প্রতিটিরই মেইন সিস্টেম মেমরি বা RAM-এ অ্যাক্সেস করার লেজিটিমেট (legitimate) বা বৈধ কারণ রয়েছে। বেসব্যান্ডকে ইনকামিং নেটওয়ার্ক প্যাকেটগুলো এমন একটি বাফারে (buffer) রাইট করতে হয় যা AP রিড বা পড়তে পারে। ওয়াই-ফাই কন্ট্রোলারকে ডাউনলোড করা ডেটা ডেলিভার করতে হয়। GPU-কে টেক্সচার ডেটা (texture data) পড়তে হয় এবং রেন্ডার করা ফ্রেমগুলো রাইট করতে হয়।

প্রাথমিক দিকের স্মার্টফোন আর্কিটেকচারগুলোতে, এই সেকেন্ডারি প্রসেসরগুলোকে **Direct Memory Access (DMA)** প্রদান করা হতো—যা ছিল এমন একটি আনরেস্ট্রিক্টেড (unrestricted) এবং হাই-স্পিড চ্যানেল, যার মাধ্যমে তারা অ্যাপ্লিকেশন প্রসেসরকে (AP) একেবারেই যুক্ত না করে মেইন সিস্টেম RAM-এর *যেকোনো* লোকেশনে সরাসরি ডেটা রিড এবং রাইট করতে পারত।

দ্য DMA অ্যাটাক ভেক্টর (The DMA Attack Vector)

Direct Memory Access বা DMA মূলত পারফরম্যান্স বা গতি বৃদ্ধির উদ্দেশ্যেই ডিজাইন করা হয়েছিল। সেকেন্ডারি প্রসেসরগুলোকে সরাসরি RAM-এ রিড এবং রাইট করার অনুমতি দিয়ে, সিস্টেম প্রতিটি ডেটা ট্রান্সফারকে CPU-এর মধ্য দিয়ে রাউট (route) করার অতিরিক্ত লোড এড়াতে পারত। কিন্তু এই আর্কিটেকচারাল শর্টকাটটি একটি ভয়াবহ সিকিউরিটি ভালনারেবিলিটির (security vulnerability) জন্ম দেয়।

যদি কোনো প্রতিপক্ষ বা অ্যাডভারসারি (adversary) একটি সেকেন্ডারি প্রসেসরকে কম্প্রোমাইজ করে ফেলে—উদাহরণস্বরূপ, ম্যালিশিয়াস রেডিও প্যাকেটের মাধ্যমে সেলুলার বেসব্যান্ডের প্রোপ্রাইটারি ফার্মওয়্যারে থাকা কোনো ভালনারেবিলিটি এক্সপ্লয়েট করার মাধ্যমে (যেমনটি অধ্যায় ২.১৩-তে বর্ণনা করা হয়েছে)—তবে সেই কম্প্রোমাইজড প্রসেসরটি অরিজিনাল হার্ডওয়্যারের সম্পূর্ণ DMA প্রিভিলেজ (privileges) ইনহেরিট (inherit) বা লাভ করে।

তখন কম্প্রোমাইজড বেসব্যান্ডটি নিচের কাজগুলো করতে পারে:

১. অ্যাপ্লিকেশন প্রসেসরের RAM রিড করা: এটি এনক্রিপশন কি, পাসওয়ার্ড এবং ডিক্রিপ্ট করা মেসেজের কন্টেন্টের জন্য সম্পূর্ণ সিস্টেম মেমরি স্ক্যান করতে পারে। এটি File-Based Encryption মাস্টার কি (master key) এক্সট্রাক্ট করে কার্যত পুরো ফাইল সিস্টেমটিকেই ডিক্রিপ্ট করে ফেলতে পারে। **২. অ্যাপ্লিকেশন প্রসেসরের RAM-এ রাইট করা:** এটি সফটওয়্যার-লেভেলের কোনো সিকিউরিটি চেক ট্রিগার (trigger) করা ছাড়াই সরাসরি কার্নেলের মেমরি স্পেসে ম্যালিশিয়াস কোড ইনজেক্ট

(inject) করতে পারে এবং একটি পারসিস্টেন্ট রুটকিট (persistent rootkit) ইনস্টল করতে পারে। SELinux এটি কোনোভাবেই শনাক্ত করতে পারে না, কারণ রাইট করার এই কাজটি অপারেটিং সিস্টেমের একদম নিচে, হার্ডওয়্যার বাস (hardware bus) লেভেলে ঘটে। ৩. **নীরবে ডেটা এক্সফিলট্রেট করা:** কম্প্রোমাইজড বেসব্যান্ডের সাথে সেলুলার নেটওয়ার্কের সরাসরি সংযোগ আগে থেকেই থাকে। ফলে এটি অ্যাপ্লিকেশন প্রসেসর বা GrapheneOS-কে বিন্দুমাত্র জানতে না দিয়েই ওভার-দ্য-এয়ার (over the air) চুরি করা ডেটা ট্রান্সমিট করতে পারে।

২০১০-এর দশকের শুরুর দিকে এটিই ছিল চরম বাস্তবতা। একটি কম্প্রোমাইজড বেসব্যান্ডের মানেই ছিল সম্পূর্ণ কম্প্রোমাইজড একটি ডিভাইস। সেখানে সফটওয়্যার সিকিউরিটি স্ট্যাকের (software security stack) কোনো প্রাসঙ্গিকতাই ছিল না।

দ্য হার্ডওয়্যার ফায়ারওয়াল: IOMMU (The Hardware Firewall: IOMMU)

আধুনিক Pixel হার্ডওয়্যার **Input-Output Memory Management Unit (IOMMU)** নামক একটি ফিজিক্যাল হার্ডওয়্যার কম্পোনেন্টের মাধ্যমে এই ভয়ংকর ভালনারেবিলিটি বা দুর্বলতাটিকে পুরোপুরি নির্মূল করে দেয়।

একটি IOMMU হলো একটি ডেডিকেটেড হার্ডওয়্যার ফায়ারওয়াল, যা সরাসরি System-on-Chip (SoC)-এর ভেতরেই বিল্ট-ইন (built-in) থাকে। এটি ফিজিক্যালি বা ভৌতভাবে প্রতিটি সেকেন্ডারি প্রসেসর এবং মেইন সিস্টেম মেমরি বাসের ঠিক মাঝখানে অবস্থান করে। সেকেন্ডারি প্রসেসরের প্রতিটি সিঙ্গেল মেমরি অ্যাক্সেস রিকোয়েস্টকে RAM-এ পৌঁছানোর আগে অবশ্যই IOMMU-এর মধ্য দিয়ে যেতে হয়।

IOMMU অপারেটিং সিস্টেম-ডিফাইনড (operating-system-defined) একটি অত্যন্ত কঠোর **Translation table** মেইনটেইন করে। এই টেবিলে একদম স্পষ্টভাবে নির্দিষ্ট করা থাকে যে, প্রতিটি সেকেন্ডারি প্রসেসর RAM-এর ঠিক কোন ক্ষুদ্র রিজিওন বা অংশে অ্যাক্সেস করার অনুমতি পাবে। বুট (boot) প্রসেসের সময় GrapheneOS এই টেবিলটি কনফিগার করে এবং IOMMU হার্ডওয়্যার সেটি ওয়্যার স্পিডে (wire speed) এনফোর্স বা প্রয়োগ করে।

যখন সেলুলার বেসব্যান্ড একটি ইনকামিং নেটওয়ার্ক প্যাকেট RAM-এ রাইট করতে চায়, তখন এটি IOMMU-এর মাধ্যমে রাইট রিকোয়েস্ট (write request) পাঠায়। IOMMU তখন ট্রান্সলেশন টেবিল চেক করে দেখে: *"বেসব্যান্ডের কি এই নির্দিষ্ট মেমরি অ্যাড্রেসে রাইট করার পারমিশন আছে?"* অ্যাড্রেসটি যদি নেটওয়ার্ক প্যাকেটের জন্য আগে থেকে নির্ধারণ করে রাখা ছোট বাফার জোনের (buffer zone) মধ্যে পড়ে, তবেই রাইট করার অনুমতি মেলে। কিন্তু বেসব্যান্ড যদি অন্য যেকোনো মেমরি রিজিওন—যেমন কার্নেল কোড, অ্যাপ্লিকেশন প্রসেসরের হিপ (heap), বা এনক্রিপশন কি স্টোরেজে—অ্যাক্সেস করার চেষ্টা করে, তবে IOMMU হার্ডওয়্যার লেভেলেই রিকোয়েস্টটি ব্লক করে দেয়।

ব্লক করা রিকোয়েস্টটি শুধু এরর (error) বা ত্রুটি রিটার্ন করেই থেমে থাকে না। IOMMU একটি হার্ডওয়্যার ফল্ট রেইজ (raise) করে, যা অ্যাপ্লিকেশন প্রসেসরে একটি ইন্টারাপ্ট (interrupt) ট্রিগার করে। GrapheneOS এই ইন্টারাপ্টটি রিসিভ করে এবং আনঅথরাইজড বা অননুমোদিত অ্যাক্সেসের চেষ্টাটি লগ (log) করতে পারে, যার মাধ্যমে সেকেন্ডারি প্রসেসরের কোনো সম্ভাব্য কম্প্রোমাইজের ব্যাপারে সিস্টেমকে সতর্ক করা যায়।

দ্য আইসোলেশন ম্যাপ (The Isolation Map)

GrapheneOS-এ IOMMU-এর ব্যবহারিক প্রভাব হলো সমস্ত সেকেন্ডারি প্রসেসরের একটি অত্যন্ত কঠোর, হার্ডওয়্যার-এনফোর্সড কম্পার্টমেন্টালাইজেশন (compartmentalization):

IOMMU হার্ডওয়্যার কম্পার্টমেন্টালাইজেশন (IOMMU Hardware Compartmentalization):

- **সেলুলার বেসব্যান্ড (Cellular Baseband):** শুধুমাত্র ছোট একটি নেটওয়ার্ক প্যাকেট বাফারে অ্যাক্সেস অনুমোদিত। কার্নেল মেমরি, অ্যাপ ডেটা এবং এনক্রিপশন কিতে অ্যাক্সেস ডিনাইড (Denied)।
- **Wi-Fi/Bluetooth Controller:** শুধুমাত্র Wi-Fi ফ্রেম বাফারে অ্যাক্সেস অনুমোদিত। কার্নেল মেমরি, অ্যাপ ডেটা এবং এনক্রিপশন কিতে অ্যাক্সেস ডিনাইড।
- **GPU:** ফ্রেমবাফার (framebuffer) এবং টেক্সচার মেমরিতে (texture memory) অ্যাক্সেস অনুমোদিত। রেন্ডারিংয়ের বাইরের যেকোনো কার্নেল কোড এবং অ্যাপ ডেটায় অ্যাক্সেস ডিনাইড।
- **ISP (Camera):** শুধুমাত্র ক্যামেরা ফ্রেম বাফারে অ্যাক্সেস অনুমোদিত। ক্যামেরা বহির্ভূত সমস্ত মেমরি রিজিওনে অ্যাক্সেস ডিনাইড।

প্রতিটি সেকেন্ডারি প্রসেসরকে একটি অত্যন্ত সংকীর্ণ এবং সুনির্দিষ্ট উদ্দেশ্য-ভিত্তিক (purpose-specific) মেমরি করিডোরে সীমাবদ্ধ রাখা হয়। এমনকি সম্পূর্ণভাবে কম্প্রোমাইজড হয়ে যাওয়া কোনো বেসব্যান্ড বা ওয়াই-ফাই চিপ আপনার Signal মেসেজ, আপনার ছবি বা আপনার এনক্রিপশন কি-এর একটি সিঙ্গেল বাইটও (single byte) পড়তে পারে না। IOMMU ফিজিক্যালি বা ভৌতভাবেই রিকোয়েস্ট রাউট (route) করতে অস্বীকার করে।

দ্য মটোরোলা পার্টনারশিপ: এক্সটেনডিং হার্ডওয়্যার ট্রাস্ট (The Motorola Partnership: Extending Hardware Trust)

২০২৫ সালে, GrapheneOS প্রথমবারের মতো নন-পিক্সেল (non-Pixel) হার্ডওয়্যারে অপারেটিং সিস্টেমটি নিয়ে আসার জন্য Motorola Mobility-এর সাথে তাদের ল্যান্ডমার্ক বা ঐতিহাসিক পার্টনারশিপের ঘোষণা দেয়। এই পার্টনারশিপের একটি অত্যন্ত ক্রিটিক্যাল বা গুরুত্বপূর্ণ শর্ত হলো, Motorola ডিভাইসগুলোকে অবশ্যই গুগলের Pixel লাইনআপের মতো একই

ক্যালিবরের (caliber) বা একই মাপের হার্ডওয়্যার আইসোলেশন প্রদান করতে হবে।

এর অর্থ হলো, অত্যন্ত সুদৃঢ় IOMMU এনফোর্সমেন্ট, Titan M2-এর সমতুল্য হার্ডওয়্যার-ব্যাকড (hardware-backed) সিকিউর এলিমেন্ট এবং যথাযথ Verified Boot কি এনরোলমেন্ট (key enrollment)। শুধুমাত্র ডিভাইস কম্প্যাটিবিলিটি (compatibility) বাড়ানোর জন্য GrapheneOS কখনোই তাদের হার্ডওয়্যার আইসোলেশন রিকোয়ারমেন্টের সাথে আপস করবে না। সিলিকনকে অবশ্যই এই মানদণ্ড পূরণ করতে হবে, অন্যথায় ডিভাইসটি সাপোর্ট করা হবে না।

ঐতিহাসিকভাবে GrapheneOS শুধুমাত্র Pixel ডিভাইসগুলোতে এক্সক্লুসিভ থাকার পেছনে এটাই হলো ফান্ডামেন্টাল কারণ—গুগল হার্ডওয়্যারের প্রতি কোনো কমার্শিয়াল বা বাণিজ্যিক পছন্দের কারণে নয়; বরং শুধুমাত্র Pixel-ই বাণিজ্যিকভাবে উপলব্ধ একমাত্র ডিভাইস ছিল যা সঠিক IOMMU কনফিগারেশন, Titan M2 সিকিউর এলিমেন্ট এবং কাস্টম AVB কি দিয়ে বুটলোডার লক করার সুবিধা প্রদান করত।

দ্য প্রিন্সিপল অব লিস্ট প্রিভিলেজ ইন সিলিকন (The Principle of Least Privilege in Silicon)

IOMMU হলো সাইবার সিকিউরিটির একটি ফান্ডামেন্টাল বা মৌলিক ডকট্রিন (doctrine)-এর হার্ডওয়্যার রূপ: **The Principle of Least Privilege** (প্রিন্সিপল অব লিস্ট প্রিভিলেজ)। সিস্টেমের প্রতিটি কম্পোনেন্ট—তা সে কোনো সফটওয়্যার প্রসেস হোক বা ফিজিক্যাল প্রসেসর—তাকে তার নির্দিষ্ট কাজটি সম্পন্ন করার জন্য ঠিক যতটুকু অ্যাক্সেস প্রয়োজন, শুধুমাত্র ততটুকু অ্যাক্সেসই প্রদান করা উচিত, এর বিন্দুমাত্র বেশি নয়।

সেলুলার বেসব্যান্ডের কাজ হলো নেটওয়ার্ক প্যাকেট ডেলিভার করা। তাকে শুধুমাত্র নেটওয়ার্ক প্যাকেট বাফারে অ্যাক্সেস দেওয়া হয়। আপনার এনক্রিপশন কি রিড করা তার কাজ নয়, তাই সিলিকন নিজেই ফিজিক্যালি তাকে সেই অ্যাক্সেস থেকে ডিনাই বা প্রত্যাখ্যান করে।

সফটওয়্যার লেয়ার (SELinux, seccomp-bpf) থেকে শুরু করে একেবারে নিচে ফিজিক্যাল হার্ডওয়্যার লেয়ার (IOMMU) পর্যন্ত Principle of Least Privilege প্রয়োগ করার মাধ্যমে, GrapheneOS এটি নিশ্চিত করে যে যেকোনো সেকেন্ডারি প্রসেসরের কোনো ভয়াবহ ফার্মওয়্যার কম্প্রোমাইজও হার্ডওয়্যার বাউন্ডারিতেই কন্টেইনড বা সীমাবদ্ধ থাকে। ড্যামেজটি কোয়ারেন্টাইনড (quarantined) হয়ে পড়ে। আর ফোর্ট্রেসটি অটলভাবে টিকে থাকে।

পর্ব ৩ এর উপসংহার: দ্য আনব্রেকঅ্যাবল ফোর্ট্রেস (Part III Conclusion: The Unbreakable Fortress)

ছয়টি লাইভ-ফায়ার সিমুলেশন (live-fire simulations) এবং তিনটি আর্কিটেকচারাল ডিপ ডাইভের (architectural deep dives) মাধ্যমে আমরা একটি সঠিকভাবে কনফিগার করা GrapheneOS ডিভাইসকে আধুনিক থ্রেট ল্যান্ডস্কেপের (threat landscape) সবচেয়ে ভয়ংকর ডিজিটাল এবং ফিজিক্যাল বা ভৌত অস্ত্রের মুখোমুখি করেছি।

- আমরা USB-C পোর্টের বিরুদ্ধে Cellebrite এবং GrayKey ফরেনসিক টার্মিনাল ডিপ্লয় করেছি। পোর্টটি সম্পূর্ণ ডেড (dead) বা নিস্তেজ ছিল।
- এনক্রিপশন কি (encryption keys) এক্সট্র্যাক্ট করার জন্য আমরা তরল নাইট্রোজেন (liquid nitrogen) দিয়ে RAM ফ্রিজ (freeze) করেছি। কিন্তু Deadman's Switch ততক্ষণে কি-গুলোকে পুরোপুরি ধ্বংস করে দিয়েছিল।
- আমরা মাল্টি-মিলিয়ন ডলার মূল্যের Pegasus-class জিরো-ক্লিক এক্সপ্লয়েট চেইন (zero-click exploit chain) লঞ্চ করেছি। বাফার ওভারফ্লো (buffer overflow) সোজা গিয়ে `hardened_malloc`-এর গার্ড পেজে (guard pages) ক্র্যাশ করেছে, Control Flow Integrity দ্বারা শেলকোড (shellcode) রিজেক্টেড বা প্রত্যাখ্যাত হয়েছে এবং seccomp-bpf কার্নেল এসক্যালেশন (kernel escalation) ডিনাই বা প্রত্যাখ্যান করেছে।
- আমরা ফেইক (fake) সেল টাওয়ার এবং ম্যালিশিয়াস Wi-Fi অ্যাক্সেস পয়েন্ট (Wi-Fi access points) সেট আপ করেছি। LTE-Only মোড ডাউনগ্রেড (downgrade) হতে অস্বীকার করেছে এবং VPN Kill Switch যেকোনো হোস্টাইল (hostile) বা বৈরী নেটওয়ার্ক স্পর্শ করার আগেই প্রতিটি বাইট (byte) এনক্রিপ্ট করে ফেলেছে।
- আমরা স্বয়ং আপডেট ডেলিভারি ইনফ্রাস্ট্রাকচারটিকেই (update delivery infrastructure) কম্প্রোমাইজ করেছি। কিন্তু Verified Boot-এর হার্ডওয়্যার-ফিউজড কি (hardware-fused keys) বিন্দুমাত্র দ্বিধা ছাড়াই সেই টেম্পারড বা দূষিত ফার্মওয়্যারটিকে (tampered firmware) রিজেক্ট করে দিয়েছে।
- আমরা একটি ট্রোজানাইজড অ্যাপ্লিকেশন (trojanized application) ইনস্টল করেছি। কিন্তু ট্রোজানটি এমন একটি শব্দভেদ্য, খালি স্যান্ডবক্সে (sandbox) অবরুদ্ধ ছিল যার কোনো ক্লিপবোর্ড অ্যাক্সেস ছিল না, ফাইল সিস্টেম ভিজিবিলিটি (visibility) ছিল না এবং কোনো নেটওয়ার্ক কানেক্টিভিটি (network connectivity) ছিল না।

প্রতিটি অ্যাটাক নিউট্রালাইজ (neutralized) বা নিষ্ক্রিয় করা হয়েছে। প্রতিটি কিল চেইন (kill chain) ভেঙে দেওয়া হয়েছে।

"আনহ্যাকঅ্যাবল" ("Unhackable") সম্পর্কিত বাস্তব সত্য

GrapheneOS-কে "আনহ্যাকঅ্যাবল" (unhackable) বলে দাবি করাটা হবে এক চরম দায়িত্বহীনতা—এবং মৌলিকভাবে একটি অসততা। মানুষের তৈরি কোনো সিস্টেমই নিখুঁতভাবে ইনভালনারেবল (invulnerable) বা অভেদ্য নয়। Android কার্নেল, Tensor system-on-chip ফার্মওয়্যার এবং সেলুলার বেসব্যান্ড স্ট্যাক (cellular baseband stack) তৈরি

করতে যে লাখ লাখ লাইনের কোড লেখা হয়েছে, তার কোথাও না কোথাও এখনও আবিষ্কার না হওয়া ভালনারেবিলিটি বা দুর্বলতা রয়ে গেছে। সেগুলোর অস্তিত্ব আছে। এবং একদিন সেগুলো ঠিকই আবিষ্কৃত হবে।

কিন্তু এখানে মূল পার্থক্যটা ভালনারেবিলিটির অস্তিত্ব থাকা নিয়ে নয়, বরং **সেগুলোকে এক্সপ্লয়েট করতে ঠিক কতটা খরচ করতে হবে** তা নিয়ে।

একটি স্ট্যান্ডার্ড কমার্শিয়াল স্মার্টফোনে, ডিভাইসটিকে কম্প্রোমাইজ করার খরচ কার্যত শূন্য (zero)। একজন ডেটা ব্রোকার প্যাসিভলি (passively) বা নীরবে আপনার তথ্য হারভেস্ট করে। একজন স্ক্রিপ্ট কিডি (script kiddie) আগে থেকেই তৈরি করে রাখা এক্সপ্লয়েট কিট (exploit kit) ডাউনলোড করে কাজ হাসিল করে। একজন স্বল্প বাজেটের স্টকার (stalker) প্লে স্টোর (Play Store) থেকে একটি কমার্শিয়াল স্পাইওয়্যার (spyware) অ্যাপ ইনস্টল করে নেয়।

কিন্তু একটি সঠিকভাবে কনফিগার করা GrapheneOS ডিভাইসে কম্প্রোমাইজ করার এই খরচটি অ্যাস্ট্রোনমিক্যাল (astronomical) বা আকাশচুম্বী। আমাদের তৈরি করা এই বহুস্তরীয় সুরক্ষাব্যবস্থা বা লেয়ার্ড ডিফেন্স (layered defenses) সফলভাবে ভেদ করতে হলে একজন অ্যাডভারসারি বা প্রতিপক্ষকে অবশ্যই নিচের কাজগুলো করতে হবে:

১. এমন কোনো কম্পোনেন্টে একটি জিরো-ডে ভালনারেবিলিটি (zero-day vulnerability) আবিষ্কার করতে হবে, যা Attack Surface Reduction দ্বারা ইতিমধ্যেই বাদ দেওয়া হয়নি। ২. একটি কার্যকরী এক্সপ্লয়েট ডেভেলপ করতে হবে যা `hardened_malloc`-এর র‍্যান্ডমাইজড হিপ লেআউট (randomized heap layout) এবং গার্ড পেজগুলোকে (guard pages) বাইপাস করতে পারে। ৩. ট্যাগ মিসম্যাচ ফল্ট (tag mismatch fault) ট্রিগার না করেই ARM MTE-এর হার্ডওয়্যার-এনফোর্সড মেমরি ট্যাগিং (hardware-enforced memory tagging) সফলভাবে নেভিগেট করতে হবে। ৪. Control Flow Integrity-র কম্পাইলার-জেনারেটেড (compiler-generated) লেজিটিমেট এক্সিকিউশন টার্গেটের হোয়াইটলিস্টকে (whitelist) পরাজিত করতে হবে। ৫. স্যান্ডবক্স (sandbox) এস্কেপ করার জন্য seccomp-bpf-এর সিস্টেম কল ফিল্টারিং এবং হার্ডেনড SELinux mandatory access controls বাইপাস করতে হবে। ৬. প্রয়োজনীয় কোড গ্যাজেটগুলো (code gadgets) খুঁজে বের করার জন্য exec spawning-এর পার-প্রসেস ASLR (per-process ASLR) অতিক্রম করতে হবে। ৭. ওপরের সমস্ত কাজ Auto-Reboot টাইমারের সীমিত সময়ের উইন্ডোর মধ্যে সফলভাবে সম্পন্ন করতে হবে—এমন একটি ডিভাইসের বিরুদ্ধে যার USB-C ডেটা লাইনগুলো বিচ্ছিন্ন, যার বেসব্যান্ড IOMMU-isolated এবং যার বুটলোডার যেকোনো আনঅথরাইজড ফার্মওয়্যারকে সরাসরি রিজেক্ট করে দেয়।

এই স্তরগুলোর প্রতিটি স্বাধীনভাবেই একেকটি দুর্ভেদ্য প্রাচীর। সবগুলোকে একত্রে মেলালে তারা এমন একটি নিশ্ছিদ্র defense-in-depth তৈরি করে, যা ভেদ করে সফলভাবে কম্প্রোমাইজ করতে গেলে বাস্তবসম্মত খরচ গিয়ে দাঁড়াবে কয়েক কোটি ডলারে—যা পৃথিবীর কেবলমাত্র সবচেয়ে অ্যাডভান্সড নেশন-স্টেট (nation-state) বা রাষ্ট্রায়ত্ত ইন্টেলিজেন্স এজেন্সিগুলোর বাজেটেই সংকুলান হওয়া সম্ভব।

আপনি নিজেকে অপরাজেয় বা ইনভিন্সিবল (invincible) করে তোলেননি। বরং আপনি নিজেকে অ্যাটাক করার জন্য অ্যাস্ট্রোনমিক্যালি এক্সপেনসিভ (astronomically expensive) বা অসম্ভব ব্যয়বহুল করে তুলেছেন। আর সাইবার ওয়ারফেয়ারের (cyber warfare) অর্থনীতিতে এটিই হলো চূড়ান্ত বিজয়।

সিকিউরিটি কোনো পণ্য বা প্রোডাক্ট নয় (Security Is Not a Product)

এই বইটির সবচেয়ে গুরুত্বপূর্ণ শিক্ষা কোনো টেকনিক্যাল বিষয় নয়। এটি মূলত একটি ফিলোসফিক্যাল (philosophical) বা দার্শনিক শিক্ষা।

কমার্শিয়াল টেকনোলজি ইন্ডাস্ট্রি দশকের পর দশক ধরে গ্রাহকদের এটি বিশ্বাস করাতে শিখিয়েছে যে সিকিউরিটি হলো একটি পণ্য। তারা আপনাকে অ্যান্টিভাইরাস সাবস্ক্রিপশন, ফায়ারওয়াল অ্যাপ্লিকেশন এবং "সিকিউর" মেসেজিং অ্যাপ বিক্রি করে। তারা আপনাকে বোঝাতে চায় যে সঠিক সফটওয়্যারটি কিনে নিলেই আপনি নিষ্ক্রিয়ভাবে নিরাপত্তা অর্জন করতে পারবেন।

এটি একটি নির্জলা মিথ্যা।

সিকিউরিটি আপনার কেনা কোনো পণ্য নয়। এটি হলো এমন একটি **আর্কিটেকচার যা আপনি যাপন করেন (architecture you inhabit)**। এটি ব্যাংকিং অ্যাপ ওপেন করার আগে প্রোফাইল সুইচ করার একটি দৈনন্দিন ডিসিপ্লিন (discipline) বা নিয়ম। এটি কোনো ওয়েদার (weather) বা আবহাওয়া অ্যাপকে লোকেশন পারমিশন দিতে অস্বীকৃতি জানানো। এটি বর্ডার এজেন্টের হাতে ডিভাইস তুলে দেওয়ার ঠিক তিন সেকেন্ড আগে রিবুট (reboot) করা। এটি নিজের সার্বভৌমত্বের মূল্য হিসেবে সচেতনভাবে ফ্রিকশন (friction) বা বাধাকে মেনে নেওয়ার একটি সুদৃঢ় সিদ্ধান্ত।

GrapheneOS গাণিতিকভাবে হার্ডেনড বা শক্তিশালী টুলস প্রদান করে। কিন্তু এই টুলগুলো এর অপারেটরকে ছাড়া সম্পূর্ণ অকার্যকর (inert)। আপনি নিজেই হলেন ডিফেন্স বা সুরক্ষার চূড়ান্ত স্তর। আপনার আচরণ, আপনার ডিসিপ্লিন এবং কনভেনিয়েন্স বা আরামদায়ক সুবিধার চেয়ে সিকিউরিটিকে অগ্রাধিকার দেওয়ার যে সদিচ্ছা—এগুলোই মূলত কিছু সফটওয়্যার টগলের সমষ্টিকে একটি দুর্ভেদ্য দুর্গে রূপান্তরিত করে।

এরপর কী? (What Comes Next)

এই বইটির মূল কারিকুলাম বা পাঠ্যক্রম এখানেই সমাপ্ত হলো। আপনি সফলভাবে 'ফাউন্ডেশন' (Foundation), 'ফোর্ট্রেস' (Fortress) এবং 'ব্যাটলফিল্ড' (Battlefield) পর্বগুলো অতিক্রম করে গ্র্যাজুয়েট (graduated) হয়েছেন। পৃথিবীর সবচেয়ে ভয়ংকর থ্রেটগুলোর বিরুদ্ধে আপনার ডিভাইসটি কীভাবে নিজেকে রক্ষা করে, তা সম্পর্কে এখন আপনার কাছে একটি গভীর, ফরেনসিক লেভেলের বোঝাপড়া বা জ্ঞান রয়েছে।

যাইহোক, GrapheneOS ইকোসিস্টেম (ecosystem) কেবল একক অপারেটরের মধ্যেই সীমাবদ্ধ নয়। এরপরের পরিশিষ্ট বা অ্যাপেন্ডিক্সগুলোতে (appendices), আমরা দুটি অ্যাডভান্সড অডিয়েন্সের (advanced audiences) জন্য বিশেষায়িত রিসোর্স সরবরাহ করব:

ডেভেলপারদের জন্য (For Developers): GrapheneOS-এর হার্ডেনড এনভায়রনমেন্টের সাথে কম্প্যাটিবল (compatible) অ্যাপ্লিকেশন তৈরি করার একটি টেকনিক্যাল রেফারেন্স (technical reference)—যা Scoped Storage-কে সম্মান করবে, Google Play Services-এর নির্ভরশীলতা ছাড়াই স্বাধীনভাবে কাজ করবে এবং `hardened_malloc`-এর কঠোর মেমরি এনফোর্সমেন্টগুলো অত্যন্ত গ্রেসফুলি (gracefully) হ্যান্ডেল করবে।

এন্টারপ্রাইজ অ্যাডমিনিস্ট্রেটরদের জন্য (For Enterprise Administrators): কর্পোরেট ডিভাইসের বিশাল ফ্লিট (fleets) জুড়ে GrapheneOS প্রোভিশন (provision) করার একটি ডিপ্লয়মেন্ট গাইড (deployment guide)—যার মধ্যে অন্তর্ভুক্ত থাকবে MDM (Mobile Device Management) কনসিডারেশনস, বাল্ক প্রোফাইল কনফিগারেশন স্ট্র্যাটেজি এবং এমন সব ইন্ডাস্ট্রির জন্য কমপ্লায়েন্স ফ্রেমওয়ার্ক (compliance frameworks) যাদের HIPAA, SOC 2, বা GDPR-গ্রেড মোবাইল সিকিউরিটি প্রয়োজন হয়।

তবে সাধারণ বা একক অপারেটরদের জন্য এই অ্যাপেন্ডিক্সগুলো পড়া বাধ্যতামূলক নয়। আপনার দুর্গ নির্মিত হয়ে গেছে। আপনার বর্ম প্রস্তুত। আপনার ডিসিপ্লিন বা শৃঙ্খলার ভিত্তি স্থাপিত হয়েছে।

আপনি এখন আর টেকনোলজির নিছক কোনো কনজিউমার (consumer) বা উপভোক্তা নন। আপনি এখন একটি গাণিতিকভাবে পরীক্ষিত, হার্ডওয়্যার-আইসোলেটেড এবং ক্রিপ্টোগ্রাফিক্যালি সিল (cryptographically sealed) করা মোবাইল প্ল্যাটফর্মের সার্বভৌম অপারেটর (sovereign operator)। সার্ভিল্যান্স ইকোনমি (surveillance economy) বা নজরদারি অর্থনীতি আজ তাদের একটি ডেটা পয়েন্ট হারিয়েছে। প্যানোপটিকনের (panopticon) চোখে আজ একটি ব্লাইন্ড স্পট (blind spot) বা অন্ধবিন্দু তৈরি হয়েছে।

অন্য পাশে আপনাকে স্বাগতম (Welcome to the other side)।

পরিশিষ্ট (Appendices)

পরিশিষ্ট ক: দ্য ডেভেলপারস ডোমেইন (Appendix A: The Developer's Domain)

পূর্ববর্তী অধ্যায়গুলো লেখা হয়েছিল মূলত অপারেটরের জন্য—সেই ব্যক্তি যিনি প্রতিদিন GrapheneOS আর্কিটেকচারটি ইনস্টল করেন, কনফিগার করেন এবং ব্যবহার করেন। একজন অপারেটরের মূল ফোকাস থাকে প্র্যাকটিক্যাল বা ব্যবহারিক বিষয়ের ওপর: *আমি এই টগলটি কীভাবে কনফিগার করব? এই থ্রেট বা হুমকি থেকে আমি কীভাবে বাঁচব?*

এই পরিশিষ্ট বা অ্যাপেন্ডিক্সে আমরা আমাদের ফোকাস পরিবর্তন করছি। এখন আমরা সরাসরি সেই সমস্ত ইঞ্জিনিয়ার, সিস্টেম অ্যাডমিনিস্ট্রেটর এবং এন্টারপ্রাইজ আর্কিটেক্টদের সাথে কথা বলছি যারা নেপথ্যে থেকে কাজ করেন। আপনাদের মূল ফোকাস থাকে কাঠামোগত বিষয়ের ওপর: *এটি কীভাবে তৈরি হয়েছে? আমি কীভাবে এটি ভেরিফাই করব? কীভাবে এটি আমার অর্গানাইজেশনের ইনফ্রাস্ট্রাকচার বা অবকাঠামোর সাথে ইন্টিগ্রেট করব?*

দ্য ওপেন-সোর্স ইমপ্যারেটিভ (The Open-Source Imperative)

কমার্শিয়াল টেকনোলজি ইন্ডাস্ট্রিতে সিকিউরিটির মার্কেটিং করা হয় মূলত বিশ্বাসের ওপর ভিত্তি করে। একটি কর্পোরেশন আপনাকে বিশ্বাস করতে বলে যে তাদের প্রোপ্রাইটারি (proprietary), ক্লোজড-সোর্স (closed-source) কোডটি নিরাপদ। আপনি সেই কোড পড়তে পারবেন না। আপনি সেটি অডিট (audit) করতে পারবেন না। আপনার ডিভাইসে চলা বাইনারি ফাইলটি সত্যিই কোম্পানির দাবি করা সোর্স কোড থেকে তৈরি কি না, তা ভেরিফাই করার কোনো উপায় আপনার নেই। আপনি মূলত একটি ব্র্যান্ডকে বিশ্বাস করছেন, কোনো গাণিতিক প্রমাণকে নয়।

GrapheneOS এই মডেলটিকে সম্পূর্ণভাবে প্রত্যাখ্যান করে।

সম্পূর্ণ GrapheneOS অপারেটিং সিস্টেম—প্রতিটি কার্নেল মডিফিকেশন, প্রতিটি হার্ডেনড মেমরি অ্যালোকেটর, প্রতিটি SELinux পলিসি এবং প্রতিটি কম্প্যাটিবিলিটি লেয়ার—পারমিসিভ লাইসেন্সিংয়ের (permissive licensing) অধীনে ওপেন-সোর্স (open-source) কোড হিসেবে প্রকাশিত। সোর্স রিপোজিটরিগুলো (source repositories) পাবলিকলি হোস্টেড, ভার্সন-কন্ট্রোলড এবং পৃথিবীর যেকোনো ব্যক্তি বা প্রতিষ্ঠানের জন্য আনরেস্ট্রিক্টেড অডিটের (unrestricted audit) জন্য উন্মুক্ত।

এই স্বচ্ছতা নিছক কোনো আদর্শগত বা আইডিওলজিকাল (ideological) বিষয় নয়। এটি সিকিউরিটির একটি ফান্ডামেন্টাল বা মৌলিক বৈশিষ্ট্য।

যখন কোনো সরকারি সংস্থা ক্লাসিফায়েড (classified) ডিভাইসগুলোতে ডিপ্লয় (deploy) করার জন্য GrapheneOS মূল্যায়ন করে, তখন তাদের ইঞ্জিনিয়ারদের কোনো নন-ডিসক্লোজার এগ্রিমেন্ট (non-disclosure agreement) সাইন করতে হয় না বা কোনো প্রাইভেট রিপোজিটরিতে অ্যাক্সেস চাইতে হয় না। তারা পাবলিক সোর্স ট্রি (public source tree) ক্লোন করে, কোডের প্রতিটি লাইন পড়ে দেখে এবং প্রজেক্টের সিকিউরিটি ক্লেইমগুলো গাণিতিকভাবে সঠিক কি না, তা স্বাধীনভাবে ভেরিফাই করে। তারা GrapheneOS ফাউন্ডেশনের সুনামের ওপর নির্ভর করে না; তারা সরাসরি কোডকে বিশ্বাস করে।

সোর্স থেকে বিল্ড করা: দ্য আলটিমেট সাপ্লাই চেইন ডিফেন্স (Building From Source: The Ultimate Supply Chain Defense)

অধ্যায় ৩.৮-এ আমরা দেখিয়েছি কীভাবে Verified Boot এবং ক্রিপ্টোগ্রাফিক সিগনেচার ভেরিফিকেশন আপডেট ডেলিভারি ইনফ্রাস্ট্রাকচারকে টার্গেট করা সাপ্লাই চেইন অ্যাটাকগুলোর (supply chain attacks) বিরুদ্ধে সুরক্ষা দেয়। কিন্তু যেসব প্রতিষ্ঠান সবচেয়ে চরম থ্রেট মডেলের অধীনে কাজ করে—যেমন ডিফেন্স কন্ট্রাক্টর, ইন্টেলিজেন্স এজেন্সি, ইনভেস্টিগেটিভ জার্নালিজম প্রতিষ্ঠান—তাদের ক্ষেত্রে এমনকি অফিসিয়াল প্রি-বিল্ট রিলিজ বাইনারিগুলোকে (pre-built release binaries) বিশ্বাস করাটাও অগ্রহণযোগ্য হতে পারে।

এর চূড়ান্ত সমাধান হিসেবে GrapheneOS বলে: **নিজেই বিল্ড করুন (build it yourself)।**

এই প্রজেক্টটি সবার জন্য উন্মুক্ত ও বিস্তারিত বিল্ড ইন্সট্রাকশন (build instructions) প্রদান করে, যা ব্যবহার করে যেকোনো ডেভেলপার একটি সাধারণ Linux ওয়ার্কস্টেশনে সম্পূর্ণ অপারেটিং সিস্টেমটিকে সোর্স থেকে কম্পাইল (compile) করতে পারেন। অপারেটিং সিস্টেমটিকে ইন-হাউস (in-house) বিল্ড করার মাধ্যমে একটি অর্গানাইজেশন সম্পূর্ণ ক্লিন এবং স্বাধীনভাবে ভেরিফায়েড একটি সাপ্লাই চেইন অর্জন করতে পারে। তারা একদম নিশ্চিতভাবে জানতে পারে তাদের ডিভাইসে ঠিক কোন কোড রান করছে, কারণ তারা নিজেরাই কোডের প্রতিটি বাইট কম্পাইল করেছে।

পরবর্তী পৃষ্ঠাগুলোতে আমরা সেইসব ডেভেলপারদের জন্য টেকনিক্যাল রেফারেন্স দেব যারা GrapheneOS-এর হার্ডেনড এনভায়রনমেন্টের সাথে কম্প্যাটিবল অ্যাপ্লিকেশন তৈরি করতে চান। এর পাশাপাশি এন্টারপ্রাইজ অ্যাডমিনিস্ট্রেটরদের জন্য ডিভাইস ফ্লিট প্রোভিশনিংয়ের (device fleets provisioning) গাইডলাইন এবং প্রজেক্টের চলমান সিকিউরিটি রিসার্চে অংশ নিতে আগ্রহী কন্ট্রিবিউটরদের জন্য রিসোর্স প্রদান করব।

দুর্গটি এখন ইনস্পেকশন বা পরিদর্শনের জন্য উন্মুক্ত। ভেতরে প্রবেশ করুন।

ক.১ GrapheneOS কম্পাইল করা: প্রি-রিকুইজিটস (A.1 Compiling GrapheneOS: Prerequisites)

সোর্স কোড থেকে একটি অপারেটিং সিস্টেম বিল্ড (build) করা কোনো সাধারণ উইকএন্ড প্রজেক্ট (weekend project) নয়। AOSP (Android Open Source Project) বিল্ড সিস্টেম হলো ওপেন-সোর্স জগতের অন্যতম ডিমান্ডিং বা চাহিদাপূর্ণ কম্পাইলেশন এনভায়রনমেন্ট (compilation environments)। এটি শত শত Git repositories সিঙ্ক্রোনাইজ (synchronize) করে, লাখ লাখ লাইনের C, C++, Java এবং Rust কোড কম্পাইল করে, Control Flow Integrity (CFI) ইনস্ট্রুমেন্টেশনের সাথে Link-Time Optimization (LTO) পারফর্ম করে এবং চূড়ান্তভাবে এমন একটি ক্রিপ্টোগ্রাফিক্যালি সাইনড ফার্মওয়্যার ইমেজ (cryptographically signed firmware image) প্রডিউস বা তৈরি করে যা সরাসরি ফিজিক্যাল হার্ডওয়্যারে ফ্ল্যাশ (flash) করার জন্য প্রস্তুত থাকে।

কোডের একটি সিঙ্গেল লাইন স্পর্শ করার আগেই, আপনার বিল্ড এনভায়রনমেন্টকে অবশ্যই কঠোর হার্ডওয়্যার এবং সফটওয়্যার রিকোয়ারমেন্ট পূরণ করতে হবে। এই প্রি-রিকুইজিটস (prerequisites) বা পূর্বশর্তগুলো পূরণে ব্যর্থ হলে বিল্ড ফেইল করবে, ইমেজ করাপ্ট (corrupted) হবে, অথবা—সবচেয়ে খারাপ ক্ষেত্রে—ডিভাইসটি ব্রিকড (bricked) বা অকেজো হয়ে যাবে।

হার্ডওয়্যার রিকোয়ারমেন্টস (Hardware Requirements)

AOSP বিল্ড সিস্টেমটি অস্বাভাবিক মাত্রায় রিসোর্স-ইন্টেনসিভ (resource-intensive)। একটি নির্ভরযোগ্য, প্রোডাকশন-গ্রেড বিল্ড ওয়ার্কস্টেশনের (production-grade build workstation) জন্য ন্যূনতম স্পেসিফিকেশনগুলো নিচে দেওয়া হলো:

প্রসেসর (Processor):

- একটি আধুনিক, মাল্টি-কোর x86_64 প্রসেসর থাকা বাধ্যতামূলক। ARM-based মেশিন (যেমন Apple Silicon Macs) সাপোর্ট করে না। বিল্ড সিস্টেমটি এক্সক্লুসিভলি শুধুমাত্র x86_64 Linux এনভায়রনমেন্টের জন্য ডিজাইন করা হয়েছে।
- একটি 8-core বা তার চেয়ে উচ্চ ক্ষমতাসম্পন্ন CPU কঠোরভাবে রেকমেন্ড করা হচ্ছে। বিল্ড সিস্টেমটি স্বয়ংক্রিয়ভাবে `NumCPU() + 2` প্যারালাল জব (parallel jobs) স্পন (spawn) করবে। যত বেশি কোর (cores), তত দ্রুত কম্পাইলেশন।

মেমরি (RAM):

- **ন্যূনতম 32 GiB।** এটি কোনো সাজেশন বা পরামর্শ নয়; এটি একটি হার্ড ফ্লোর (hard floor) বা আবশ্যকীয় শর্ত। বিল্ড প্রসেসের সবচেয়ে মেমরি-ইন্টেনসিভ ফেজ বা ধাপগুলো হলো Linux কার্নেল এবং Vanadium (Chromium)-এর জন্য Link-Time Optimization (LTO) পাসগুলো। CFI-এর সাথে LTO যুক্ত হয়ে লিংকিং ফেজে (linking phase) এনরমাস (enormous) বা বিশাল পিক মেমরি ডিমান্ড (peak memory demands) তৈরি করে। এই পিক চলাকালীন আপনার সিস্টেমে যদি RAM শেষ হয়ে যায়, তবে OOM (Out of Memory) handler দ্বারা লিংকারটিকে কিল (killed) করে দেওয়া হবে এবং বিল্ডটি নীরবে ফেইল করবে।
- ডেভেলপমেন্ট ওয়ার্কস্টেশনগুলোর জন্য 64 GiB রেকমেন্ড করা হয়, যেখানে আপনি অন্যান্য টাস্কের পাশাপাশি ঘন ঘন ইনক্রিমেন্টাল বিল্ড (incremental builds) সম্পাদন করবেন।

স্টোরেজ (Storage):

- **ন্যূনতম 250 GiB ফাস্ট স্টোরেজ (fast storage) প্রয়োজন।** এর ব্রেকডাউন নিচে দেওয়া হলো:
  - একটি স্ট্যান্ডার্ড সোর্স ট্রি সিঙ্কের (source tree sync) জন্য ~136 GiB (Git হিস্ট্রিসহ)। ফুল হিস্ট্রি ছাড়া একটি লাইটওয়েট সিঙ্কের (lightweight sync) জন্য প্রায় 90 GiB প্রয়োজন হয়।
  - বিল্ড আউটপুট ডিরেক্টরির (`out/`) জন্য অতিরিক্ত ~100 GiB ফ্রি স্পেস, যেখানে ইন্টারমিডিয়েট কম্পাইলেশন আর্টিফ্যাক্টস, অবজেক্ট ফাইল এবং ফাইনাল সাইনড ইমেজগুলো থাকে।
- **NVMe SSD স্টোরেজ কঠোরভাবে রেকমেন্ড করা হচ্ছে।** বিল্ড সিস্টেমটি হাজার হাজার ফাইলের ওপর বিপুল পরিমাণ ছোট, র‍্যান্ডম রিড/রাইট (random read/write) অপারেশন পারফর্ম করে। একটি ট্র্যাডিশনাল স্পিনিং হার্ড ড্রাইভ (HDD) ব্যবহার করলে বিল্ড টাইম কয়েক ঘণ্টা ছাড়িয়ে যাবে। একটি হাই-স্পিড NVMe SSD একটি ওয়েল-ইকুইপড (well-equipped) ওয়ার্কস্টেশনে ফুল বিল্ডের সময়কে দুই ঘণ্টারও নিচে নামিয়ে আনতে পারে।

অপারেটিং সিস্টেম রিকোয়ারমেন্টস (Operating System Requirements)

GrapheneOS অফিসিয়ালি নিচের Linux ডিস্ট্রিবিউশনগুলোতে বিল্ড সাপোর্ট করে:

- **Arch Linux**
- **Debian Bookworm (12)**
- **Ubuntu 24.04 LTS অথবা Ubuntu 24.10**

অন্যান্য ডিস্ট্রিবিউশনগুলো কাজ করতে পারে, তবে সেগুলো অফিসিয়ালি টেস্টেড বা সাপোর্টেড নয়। macOS এবং Windows সমর্থিত নয়। আপনাকে অবশ্যই একটি নেটিভ Linux এনভায়রনমেন্টে অপারেট করতে হবে (Windows Subsystem for Linux নয়)।

সফটওয়্যার ডিপেন্ডেন্সিজ (Software Dependencies)

আপনার Linux এনভায়রনমেন্ট প্রতিষ্ঠিত হয়ে গেলে, বেইজলাইন বিল্ড ডিপেন্ডেন্সিগুলো (baseline build dependencies) ইনস্টল করুন:

```
apt install repo yarnpkg zip rsync
```

AOSP বিল্ড সিস্টেমটি মূলত সেলফ-কন্টেইনড (self-contained)—রিপ্রোডিউসিবল বিল্ড (reproducible builds) নিশ্চিত করার জন্য এটি নিজস্ব প্রি-বিল্ট টুলচেইন (prebuilt toolchain) (যেমন- কম্পাইলার, লিংকার এবং ইউটিলিটি) শিপ (ship) করে, যা হোস্ট সিস্টেমের টুল ভার্সনগুলোর ওপর নির্ভরশীল নয়। তবে, বেশ কয়েকটি হোস্ট-লেভেল ডিপেন্ডেন্সিকে (host-level dependencies) আলাদাভাবে ইনস্টল করতে হবে:

- **কোর (Core):** `repo`, `python3`, `git`, `gnupg`, `openssh`
- **বিল্ড (Build):** `diff` (diffutils), `freetype2`, `fontconfig`, `hostname` (inetutils), `openssl`, `rsync`, `unzip`, `zip`
- **ভেন্ডর এক্সট্র্যাকশন (Vendor Extraction):** Node.js 24 LTS, `yarn`
- **Vanadium (Chromium):** `gperf`, 32-bit `glibc`, 32-bit GCC রানটাইম লাইব্রেরি
- **সাইনিং (Signing):** `ssh-keygen` (OpenSSH)

নিশ্চিত করুন যে আপনার `PATH`-এ `/sbin`, `/usr/sbin`, এবং `/usr/local/sbin` অন্তর্ভুক্ত রয়েছে, কারণ বিল্ড স্ক্রিপ্টগুলোর জন্য প্রয়োজনীয় অনেক সিস্টেম অ্যাডমিনিস্ট্রেশন কমান্ড এই ডিরেক্টরিগুলোতে অবস্থান করে।

রিলিজ সাইনিং কি জেনারেট করা (Generating Release Signing Keys)

সম্পূর্ণ বিল্ড প্রসেসে এটি হলো সিঙ্গেল মোস্ট ক্রিটিক্যাল বা সবচেয়ে গুরুত্বপূর্ণ ধাপ। আপনার রিলিজ সাইনিং কি (release signing keys) হলো আপনার কাস্টম বিল্ডের ক্রিপ্টোগ্রাফিক আইডেন্টিটি বা পরিচয়। আপনি যখন বুটলোডার (bootloader) লক করবেন, তখন এই কি-গুলোই Titan M2-এর Verified Boot chain-এ ফিউজ (fused) হবে। যদি এই কি-গুলো কম্প্রোমাইজ হয়ে যায়, তবে আপনার সম্পূর্ণ সাপ্লাই চেইন কম্প্রোমাইজ হয়ে যাবে।

দ্য সিকিউরিটি প্রোটোকল (The Security Protocol): **১. একটি এয়ার-গ্যাপড মেশিনে (air-gapped machine) কি জেনারেট করুন।** যে ওয়ার্কস্টেশনে আপনি সাইনিং কি জেনারেট এবং স্টোর করবেন, সেটিকে আদর্শগতভাবে কখনই ইন্টারনেটের সাথে যুক্ত করা উচিত নয়। এটি রিমোট কি থেফটের (remote key theft) বা দূর থেকে কি চুরি হওয়ার

সম্ভাবনাকে পুরোপুরি নির্মূল করে। ২. **বিশ্রামরত অবস্থায় কি-গুলো এনক্রিপ্ট করুন (Encrypt the keys at rest)।** GrapheneOS `script/encrypt-keys` প্রদান করে, যা কি ডেরিভেশনের (key derivation) জন্য `scrypt` এবং সাইফার (cipher) হিসেবে AES-256 ব্যবহার করে আপনার সাইনিং কি-গুলোকে এনক্রিপ্ট করে। আপনাকে একটি স্ট্রং পাসফ্রেজ (strong passphrase) দিতে বলা হবে। ৩. **সাইনিংয়ের সময় সোয়াপ (swap) ডিজেবল করুন।** আপনার বিল্ড মেশিনে যদি swap space ব্যবহৃত হয়, তবে সাইনিং প্রসেস শেষ হওয়ার পরও এনক্রিপ্টেড কি মেটেরিয়াল ডিস্কের সোয়াপ পার্টিশনে (swap partition) রাইট হয়ে থেকে যেতে পারে। কি জেনারেশন এবং সাইনিং ফেজ চলাকালীন swap সম্পূর্ণভাবে ডিজেবল করুন। যদি swap অ্যাক্টিভ রাখতেই হয়, তবে নিশ্চিত করুন যে এটি কোনো এফিমিরাল কি (ephemeral key) দিয়ে এনক্রিপ্ট করা রয়েছে। ৪. **প্রতিটি ডিভাইস ভ্যারিয়েন্টের জন্য ইউনিক কি (unique keys) জেনারেট করুন।** ভিন্ন ভিন্ন Pixel মডেলের মধ্যে একই কি রিইউজ (reuse) করবেন লিংক। প্রতিটি ডিভাইসের কোডনেমের (যেমন- Pixel 8 Pro এর জন্য `husky`, Pixel 9a এর জন্য `tegu`) নিজস্ব ডেডিকেটেড কি সেট (dedicated key set) থাকা উচিত।

কি জেনারেশন প্রসেসটি একাধিক কি পেয়ার (key pairs) তৈরি করে: `releasekey`, `platform`, `shared`, `media`, `networkstack`, `bluetooth`, `sdk_sandbox` এবং অত্যন্ত ক্রিটিক্যাল `avb.pem` (Android Verified Boot key)। বুটলোডার লক করার সময় `avb.pem` কি থেকে এক্সট্র্যাক্ট করা `avb_pkmd.bin` ফাইলটিই মূলত ডিভাইসের secure element-এ এনরোল (enroll) করা হয়।

এই কি-গুলোকে নিজের জীবনের মতো রক্ষা করুন। এগুলোই হলো আপনার সম্পূর্ণ ডিপ্লয়মেন্টের ম্যাথমেটিকাল রুট অব ট্রাস্ট (mathematical root of trust)।

ক.২ GrapheneOS কম্পাইল করা: দ্য বিল্ড প্রসেস (A.2 Compiling GrapheneOS: The Build Process)

আপনার হার্ডওয়্যার প্রস্তুত হওয়ার পর, Linux এনভায়রনমেন্ট কনফিগার করা হলে এবং একটি air-gapped মেশিনে সাইনিং কি (signing keys) জেনারেট করার পর, আপনি সোর্স (source) থেকে GrapheneOS কম্পাইল (compile) করার জন্য প্রস্তুত।

এই পরিশিষ্টে একটি প্রোডাকশন-রেডি (production-ready), সাইনড ফ্যাক্টরি ইমেজ (signed factory image) বিল্ড করার জন্য একটি হাই-লেভেল, স্টেপ-বাই-স্টেপ টার্মিনাল গাইড প্রদান করা হয়েছে। সম্পূর্ণ প্রসেসটি পাঁচটি ফেজ (phases) বা ধাপে বিভক্ত: সোর্স সিঙ্ক্রোনাইজেশন (source synchronization), ভেন্ডর ফাইল এক্সট্র্যাকশন (vendor file extraction), এনভায়রনমেন্ট কনফিগারেশন (environment configuration), কম্পাইলেশন (compilation) এবং সাইনিং (signing)।

পুরো গাইড জুড়ে আমরা টার্গেট ডিভাইস হিসেবে Pixel 9a (কোডনেম `tegu`) ব্যবহার করব। অ্যাপেন্ডিক্স এ.১-এ দেওয়া তালিকা অনুযায়ী `tegu`-এর জায়গায় আপনার নির্দিষ্ট ডিভাইসের কোডনেমটি বসিয়ে নিন।

ফেজ ১: সোর্স ট্রি ডাউনলোড করা (Phase 1: Downloading the Source Tree)

GrapheneOS তাদের সোর্স কোড শত শত Git repositories-এ মেইনটেইন করে, যা গুগলের `repo` টুল দ্বারা ম্যানেজ করা হয়। `repo` টুল একটি manifest file রিড করে যা নির্দিষ্ট কোনো রিলিজের (release) জন্য প্রতিটি রিপোজিটরি, তার ব্রাঞ্চ (branch) এবং নির্দিষ্ট commit hash ডিফাইন করে দেয়।

অপশন এ — ডেভেলপমেন্ট ব্রাঞ্চ সিঙ্ক করা (Option A — Syncing the Development Branch):

ডেভেলপমেন্ট ব্রাঞ্চ (development branch) সর্বশেষ অ্যাক্টিভ কাজের ট্র্যাক রাখে। আপনি যদি কোনো প্যাচ (patches) কন্ট্রিবিউট করতে চান বা ব্লিডিং-এজ (bleeding-edge) ফিচারগুলো টেস্ট করতে চান, তবে এটি ব্যবহার করুন।

```
mkdir grapheneos-16-qpr2
cd grapheneos-16-qpr2
repo init -u https://github.com/GrapheneOS/platform_manifest.git -b 16-qpr2
repo sync -j8
```

অপশন বি — স্টেবল রিলিজ ট্যাগ সিঙ্ক করা (প্রোডাকশনের জন্য রেকমেন্ডেড) (Option B — Syncing a Stable Release Tag (Recommended for Production)):

প্রোডাকশন বিল্ডের জন্য, আপনার সর্বদা একটি নির্দিষ্ট এবং টেস্টেড রিলিজ ট্যাগকে (Stable Release Tag) টার্গেট করা উচিত। আপনার ডিভাইসের জন্য সর্বশেষ স্টেবল ট্যাগটি খুঁজে পেতে অফিসিয়াল GrapheneOS Releases পেজটি ভিজিট করুন।

```
mkdir grapheneos-TAG_NAME
cd grapheneos-TAG_NAME
repo init -u https://github.com/GrapheneOS/platform_manifest.git -b
refs/tags/TAG_NAME
```

সিঙ্ক (sync) সম্পন্ন করার আগে, আপনি যে কোনো টেম্পারড (tampered) কোড নয় বরং অথেন্টিক (authentic) সোর্স কোড থেকে বিল্ড করছেন তা নিশ্চিত করার জন্য ম্যানিফেস্টের ক্রিপ্টোগ্রাফিক সিগনেচারটি (cryptographic signature) ভেরিফাই করুন:

```
curl https://grapheneos.org/allowed_signers >
~/.ssh/grapheneos_allowed_signers
cd .repo/manifests
git config gpg.ssh.allowedSignersFile ~/.ssh/grapheneos_allowed_signers
git verify-tag $(git describe)
cd ../..
```

যদি ভেরিফিকেশন সফল হয়, তবে ডাউনলোড সম্পন্ন করুন:

```
repo sync -j8
```

এই সিঙ্ক্রোনাইজেশনটি প্রায় 90-136 GiB সোর্স কোড ডাউনলোড করবে, যা নির্ভর করে আপনি সম্পূর্ণ Git হিস্ট্রি অন্তর্ভুক্ত করছেন কি না তার ওপর। সিঙ্ক চলার সময় যদি আপনার নেটওয়ার্ক কানেকশন ড্রপ করে, তবে সিম্পলি আবার `repo sync -j8` রান করুন। এই টুলটি যেকোনো ইন্টারাপশন (interruption) বা বাধা থেকে খুব গ্রেসফুলি (gracefully) রিজিউম (resume) করার জন্য ডিজাইন করা হয়েছে।

ফেজ ২: ভেন্ডর ফাইল এক্সট্র্যাক্ট করা (Phase 2: Extracting Vendor Files)

আধুনিক Pixel ডিভাইসগুলোর জন্য প্রোপ্রাইটারি ভেন্ডর ফাইল (proprietary vendor files) প্রয়োজন হয়—যেমন সেলুলার বেসব্যান্ডের জন্য বাইনারি ফার্মওয়্যার ব্লব (binary firmware blobs), GPU driver libraries, ক্যামেরা ISP এবং অন্যান্য হার্ডওয়্যার-স্পেসিফিক কম্পোনেন্ট। এই ফাইলগুলো ওপেন-সোর্স নয় (এগুলো হার্ডওয়্যার ম্যানুফ্যাকচারার দ্বারা সাইন করা থাকে এবং secure boot-এর সময় ভেরিফাই করা হয়), তবে ডিভাইসটি কাজ করার জন্য এগুলো আবশ্যক।

অফিসিয়াল ফ্যাক্টরি ইমেজ থেকে স্বয়ংক্রিয়ভাবে এই ভেন্ডর ফাইলগুলো এক্সট্র্যাক্ট (extract) করার জন্য GrapheneOS `adevtool` নামক একটি টুল সরবরাহ করে।

প্রথমে, `adevtool` এনভায়রনমেন্ট ইনিশিয়ালাইজ করুন (এটি কেবল একবার রান করলেই হবে):

```
yarn --cwd vendor/adevtool/ install
```

এরপর, আপনার টার্গেট ডিভাইসের জন্য ভেন্ডর ফাইলগুলো ডাউনলোড, এক্সট্র্যাক্ট এবং প্রস্তুত করুন:

```
adevtool generate-all -d tegu
```

`tegu` এর জায়গায় আপনার ডিভাইসের কোডনেমটি বসিয়ে নিন। এই কমান্ডটি গুগলের অফিসিয়াল ফ্যাক্টরি ইমেজটি ডাউনলোড করে, সেখান থেকে প্রোপ্রাইটারি ব্লবগুলো (proprietary blobs) এক্সট্র্যাক্ট করে এবং সোর্স ট্রির ভেতরের সঠিক ডিরেক্টরি স্ট্রাকচারে (directory structure) প্লেস করে।

Debian সিস্টেমের ক্ষেত্রে একটি নোট: `yarn` কমান্ডটি `cmdtest` প্যাকেজের একটি বাইনারির সাথে কনফ্লিক্ট (conflict) করতে পারে। যদি কোনো এরর বা ত্রুটির সম্মুখীন হন, তবে `yarn`-এর পরিবর্তে `yarnpkg` ব্যবহার করুন।

ফেজ ৩: বিল্ড এনভায়রনমেন্ট কনফিগার করা (Phase 3: Configuring the Build Environment)

AOSP build system-এর জন্য একটি শেল স্ক্রিপ্টের (shell script) মাধ্যমে ইনিশিয়ালাইজেশন প্রয়োজন হয়, যা toolchain paths, build functions এবং environment variables সেট আপ করে। এটি অবশ্যই `bash` বা `zsh` থেকে এক্সিকিউট করতে হবে—অন্যান্য শেলগুলো এর সাথে কম্প্যাটিবল নয়।

```
source build/envsetup.sh
```

এরপর, আপনার বিল্ড টার্গেট সিলেক্ট করুন। টার্গেটটি `DEVICE-cur-user` ফরম্যাটে উল্লেখ করতে হয়:

```
lunch tegu-cur-user
```

এই `user` ভ্যারিয়েন্টটি একটি প্রোডাকশন-গ্রেড, সিকিউরিটি-হার্ডেনড বিল্ড প্রডিউস করে। ডেভেলপমেন্ট এবং ডিবাগিংয়ের উদ্দেশ্যে আপনি এর বদলে `userdebug` ব্যবহার করতে পারেন, তবে **কখনোই কোনো `userdebug` বিল্ড প্রোডাকশনে ডিপ্লয় করবেন না**—এটি অতিরিক্ত ডিবাগিং সারফেস (debugging surfaces) এনাবল (enable) করে সিস্টেমের সিকিউরিটি পোসচারকে (security posture) মারাত্মকভাবে দুর্বল করে দেয়।

আপনি যদি GrapheneOS Updater অ্যাপ্লিকেশনটি অন্তর্ভুক্ত করতে চান (যাতে আপনার নিজস্ব ইনফ্রাস্ট্রাকচার থেকে over-the-air updates সাপোর্ট করে), তবে অফিসিয়াল বিল্ড ফ্ল্যাগটি (official build flag) সেট করুন:

```
export OFFICIAL_BUILD=true
```

ক্রিটিক্যাল ওয়ার্নিং (Critical Warning): যদি আপনি `OFFICIAL_BUILD=true` সেট করেন, তবে আপনাকে **অবশ্যই** `packages/apps/Updater/res/values/config.xml` ফাইলে থাকা আপডেট সার্ভারের (update server) URL পরিবর্তন করে আপনার নিজস্ব আপডেট সার্ভারের দিকে পয়েন্ট করতে হবে। ডিফল্ট URL রেখে দিলে, আপনার ডিভাইসগুলো বারবার ফাউন্ডেশনের কি (keys) দ্বারা সাইন করা অফিসিয়াল GrapheneOS আপডেট ডাউনলোড করবে, সেগুলোকে করাপ্টেড হিসেবে ডিটেক্ট করবে (কারণ সেগুলো আপনার সাইনিং কি-এর সাথে মিলবে না), সেগুলোকে ডিলিট করবে এবং একটি ইনফিনিট লুপে (infinite loop) বারবার রি-ডাউনলোড করতে থাকবে—যা কার্যত অফিসিয়াল ইনফ্রাস্ট্রাকচারের বিরুদ্ধে একটি ডিনায়াল-অব-সার্ভিস অ্যাটাক (denial-of-service attack) পারফর্ম করবে।

ফেজ ৪: কম্পাইলেশন (Phase 4: Compilation)

একটি প্রোডাকশন বিল্ড শুরু করার আগে, নিশ্চিত করুন যে পূর্ববর্তী বিল্ডের কোনো অবশিষ্ট অংশ (remnants) যাতে বর্তমান আউটপুটকে দূষিত বা কন্টামিনেট (contaminate) করতে না পারে:

```
rm -r out
```

বিল্ড ইনিশিয়েট (initiate) করুন। ডিভাইসের জেনারেশনের ওপর ভিত্তি করে নির্দিষ্ট বিল্ড কমান্ডটি সামান্য আলাদা হতে পারে:

Pixel 7 থেকে Pixel 9 সিরিজের জন্য:

```
m vendorbootimage vendorkernelbootimage target-files-package
```

Pixel 6 সিরিজের জন্য:

```
m vendorbootimage target-files-package
```

Pixel 10 সিরিজ এবং ইমুলেটর (emulator) টার্গেটের জন্য, বর্তমান কমান্ড সম্পর্কে জানতে সর্বশেষ অফিসিয়াল বিল্ড ডকুমেন্টেশন চেক করুন।

এই `m` কমান্ডটি হলো আন্ডারলাইং বিল্ড সিস্টেমের চারপাশের একটি র‍্যাপার (wrapper)। এটি স্বয়ংক্রিয়ভাবে উপলব্ধ সমস্ত CPU cores জুড়ে কম্পাইলেশনকে প্যারালালাইজ (parallelize) করবে (ডিফল্টভাবে `NumCPU() + 2` জবস)। আপনি চাইলে `-j` ফ্ল্যাগ ব্যবহার করে ম্যানুয়ালি প্যারালালিজম কন্ট্রোল করতে পারেন:

```
m -j8 vendorbootimage vendorkernelbootimage target-files-package
```

একটি ওয়েল-ইকুইপড ওয়ার্কস্টেশনে (16-core CPU, 64 GiB RAM, NVMe SSD) একটি ফুল, ক্লিন প্রোডাকশন বিল্ড সম্পন্ন হতে সাধারণত ১.৫ থেকে ৩ ঘণ্টা সময় লাগে। কম শক্তিশালী হার্ডওয়্যারে বিল্ড টাইম উল্লেখযোগ্য হারে বেশি লাগতে পারে। কার্নেল এবং Vanadium (Chromium)-এর জন্য LTO এবং CFI পাসগুলো (passes) হলো সবচেয়ে বেশি সময়সাপেক্ষ ফেজ।

ফেজ ৫: সাইনিং এবং ফ্যাক্টরি ইমেজ জেনারেট করা (Phase 5: Signing and Generating Factory Images)

কম্পাইলেশন শেষ হওয়ার পর, আপনাকে অবশ্যই আপনার প্রাইভেট রিলিজ কি (private release keys) (যা অ্যাপেন্ডিক্স এ.১-এ জেনারেট করা হয়েছে) দিয়ে বিল্ডটিকে সাইন (sign) করতে হবে এবং আউটপুটটিকে ফ্ল্যাশেবল ফ্যাক্টরি ইমেজ (flashable factory images) ও OTA update packages হিসেবে প্যাকেজ করতে হবে।

প্রথমে, OTA tools package বিল্ড করুন:

```
m otatools-package
```

এরপর, বিল্ড আর্টিফ্যাক্টসগুলো (build artifacts) ফাইনালাইজ (finalize) করুন:

```
script/finalize.sh
```

অবশেষে, সাইনড রিলিজ জেনারেট করুন। এই কমান্ডটি র (raw) বিল্ড আউটপুট গ্রহণ করে, আপনার প্রাইভেট কি দিয়ে প্রতিটি কম্পোনেন্টকে (APKs, dm-verity trees, boot image এবং vbmeta partition) রি-সাইন (re-signs) করে এবং ফাইনাল ফ্যাক্টরি ইমেজ জিপ ও OTA update package প্রডিউস করে:

```
script/generate-release.sh tegu BUILD_NUMBER
```

`BUILD_NUMBER`-এর জায়গায় বিল্ড আইডেন্টিফায়ারটি (build identifier) বসিয়ে দিন (সাধারণত তারিখ-ভিত্তিক একটি স্ট্রিং যেমন `2025110800`)। সাইন করা ফ্যাক্টরি ইমেজ এবং আপডেট প্যাকেজগুলো নিচের লোকেশনে আউটপুট হবে:

```
releases/BUILD_NUMBER/release-tegu-BUILD_NUMBER/
```

এই ডিরেক্টরিতে সম্পূর্ণ, সাইনড ফ্যাক্টরি ইমেজ সেটটি থাকবে, যা এই বইয়ের ১.৫ অধ্যায়ে বর্ণিত স্ট্যান্ডার্ড `fastboot` প্রসিডিউর ব্যবহার করে কোনো ডিভাইসে ফ্ল্যাশ করা যায়।

ফেজ ৬: ডেল্টা আপডেট জেনারেট করা (ঐচ্ছিক) (Phase 6: Generating Delta Updates (Optional))

যেসব প্রতিষ্ঠান একাধিক ডিভাইসের একটি বিশাল ফ্লিট (fleet) জুড়ে GrapheneOS ডিপ্লয় করে, তাদের জন্য সম্পূর্ণ OTA updates (যার মধ্যে পুরো অপারেটিং সিস্টেম থাকে) অত্যন্ত ব্যান্ডউইথ-ইন্টেনসিভ (bandwidth-intensive) বা ডেটা-খরচসাপেক্ষ হয়ে দাঁড়ায়। GrapheneOS অত্যন্ত কার্যকরী **ডেল্টা (ইনক্রিমেন্টাল) আপডেট (Delta or incremental updates)** সাপোর্ট করে, যা শুধুমাত্র দুটি রিলিজের মাঝখানের বাইনারি পার্থক্যগুলোই (binary differences) শিপ করে।

দুটি সাইনড রিলিজের মধ্যে একটি ডেল্টা আপডেট (delta update) জেনারেট করতে:

```
script/generate-delta.sh tegu 2025110600 2025110800
```

এর জন্য প্রয়োজন যে, সোর্স এবং টার্গেট—উভয় রিলিজের সাইনড `target_files` আর্কাইভগুলো আপনার `releases/` ডিরেক্টরিতে সংরক্ষিত থাকতে হবে। এরপর ডেল্টা প্যাকেজটি ফুল আপডেট প্যাকেজ এবং channel metadata-এর পাশাপাশি আপনার আপডেট সার্ভারে আপলোড করা হয়। GrapheneOS Updater ক্লায়েন্ট স্বয়ংক্রিয়ভাবে ডেল্টা আপডেটটিকে ডিটেক্ট (detect) করবে এবং সেটিকে অগ্রাধিকার দেবে।

দ্য চেইন অব ট্রাস্ট (The Chain of Trust)

আপনি যখন নিজে থেকে কম্পাইল করা এই ইমেজটিকে একটি Pixel ডিভাইসে ফ্ল্যাশ করেন এবং বুটলোডার লক করে দেন, তখন Titan M2 secure element আপনার `avb_pkmd.bin` পাবলিক কি-কে (public key) এনরোল করে নেয়। ঠিক সেই মুহূর্ত থেকে, প্রতিটি বুট সিকোয়েন্স (boot sequence) আপনার কি-এর বিপরীতে ক্রিপ্টোগ্রাফিক্যালি ভেরিফাই করা হয়—গুগলের কি দিয়ে নয় এবং GrapheneOS ফাউন্ডেশনের কি দিয়েও নয়।

এর মাধ্যমে আপনি চূড়ান্ত সাপ্লাই চেইন গ্যারান্টি (supply chain guarantee) অর্জন করেছেন। ডিভাইসে চলা প্রতিটি বাইট কোড অডিটেবল সোর্স (auditable source) থেকে কম্পাইল করা হয়েছে, এমন কি দিয়ে সাইন করা হয়েছে যা কখনোই আপনার air-gapped মেশিন থেকে বাইরে যায়নি, এবং টেম্পার-রেজিস্ট্যান্ট (tamper-resistant) হার্ডওয়্যার দ্বারা ভেরিফাই করা হয়েছে। এই চেইন অব ট্রাস্ট (chain of trust) শুধুমাত্র আপনার থেকেই শুরু হয় এবং আপনার কাছেই শেষ হয়।

একটি সিঙ্গেল ডিভাইসের জন্য বিল্ড প্রসেস মাস্টার (master) করার পর, এর পরবর্তী যৌক্তিক চ্যালেঞ্জটি হলো স্কেল (scale) করা। পরিশিষ্ট বি (Appendix B)-তে, আমরা সম্পূর্ণ অর্গানাইজেশন জুড়ে GrapheneOS ডিপ্লয় করার অপারেশনাল বাস্তবতা নিয়ে আলোচনা করব—ফ্লিট প্রোভিশনিং (provisioning fleets), রিমোটলি পলিসি এনফোর্স করা এবং ক্লাউডের (cloud) কাছে ডেটা সারেন্ডার না করেই এনক্রিপ্টেড ব্যাকআপ ম্যানেজ করা।

পরিশিষ্ট খ: এন্টারপ্রাইজ ডিপ্লয়মেন্ট এবং MDM (Appendix B: Enterprise Deployment & MDM)

পূর্ববর্তী অধ্যায় এবং পরিশিষ্টগুলোতে আমরা প্রধানত একক অপারেটরের (individual operator) ওপর ফোকাস করেছি—যেখানে একজন ব্যক্তি তার একটিমাত্র ডিভাইসকে সুরক্ষিত করছেন। তবে, GrapheneOS বর্তমানে ব্যাপক হারে প্রাতিষ্ঠানিক বা অর্গানাইজেশনাল স্কেলে (organizational scale) ডিপ্লয় (deploy) করা হচ্ছে। নিউজ-রুম (newsrooms), নন-গভর্নমেন্টাল অর্গানাইজেশন বা NGOs, সংবেদনশীল লিটিগেশন (litigation) হ্যান্ডেল করা ল ফার্ম, HIPAA দ্বারা আবদ্ধ মেডিকেল প্রতিষ্ঠান এবং ইন্ডাস্ট্রিয়াল এসপিওনাজ (industrial espionage) বা শিল্প গুপ্তচরবৃত্তির বিরুদ্ধে ডিফেন্স তৈরি করা কর্পোরেট সিকিউরিটি টিমগুলো এটি ব্যবহার করছে।

একাধিক ডিভাইসের একটি বিশাল ফ্লিট (fleet) জুড়ে একটি হার্ডেনড (hardened) মোবাইল অপারেটিং সিস্টেম ডিপ্লয় করা বেশ কিছু স্বতন্ত্র চ্যালেঞ্জ তৈরি করে। অ্যাডমিনিস্ট্রেটরদের অবশ্যই রিমোটলি (remotely) ডিভাইসগুলো প্রোভিশন (provision) করতে হয়, ইউনিফর্ম সিকিউরিটি পলিসি (uniform security policies) এনফোর্স করতে হয়, অ্যাপ্লিকেশন ডিস্ট্রিবিউশন (application distribution) ম্যানেজ করতে হয় এবং এনক্রিপ্টেড ব্যাকআপ মেইনটেইন করতে হয়—যার সবকিছুই সম্পন্ন করতে হয় গুগল ইকোসিস্টেমের (Google ecosystem) ওপর বিন্দুমাত্র নির্ভর না করে।

অ্যান্ড্রয়েড এন্টারপ্রাইজ: দ্য MDM ফ্রেমওয়ার্ক (Android Enterprise: The MDM Framework)

একটি প্রচলিত ভুল ধারণা হলো, GrapheneOS যেহেতু একটি ডি-গুগলড (de-Googled) অপারেটিং সিস্টেম, তাই এটি এন্টারপ্রাইজ ডিভাইস ম্যানেজমেন্টের সাথে কম্প্যাটিবল (compatible) নয়। এটি সম্পূর্ণ ভুল।

GrapheneOS সম্পূর্ণরূপে **Android Enterprise** স্ট্যান্ডার্ড সাপোর্ট করে—যা হলো Mobile Device Management (MDM)-এর জন্য একটি ইন্ডাস্ট্রি-ওয়াইড (industry-wide) ফ্রেমওয়ার্ক। Android Enterprise কোনো গুগলের নিজস্ব সার্ভিস নয়; এটি Android Open Source Project-এর ভেতরে বিল্ট-ইন (built-in) একটি ওপেন স্পেসিফিকেশন (open specification)। যেহেতু GrapheneOS সরাসরি AOSP-এর ওপর ভিত্তি করে তৈরি, তাই এটি Android Enterprise APIs-এর সাথে সম্পূর্ণ কম্প্যাটিবিলিটি ইনহেরিট (inherit) বা উত্তরাধিকারসূত্রে লাভ করে।

এর অর্থ হলো, একটি অর্গানাইজেশন তাদের GrapheneOS ডিভাইসগুলোকে যেকোনো স্ট্যান্ডার্ডস-কমপ্লায়েন্ট (standards-compliant) MDM প্ল্যাটফর্মে এনরোল (enroll) করতে পারে এবং রিমোটলি পুরো ফ্লিটকে অ্যাডমিনিস্টার (administer) বা পরিচালনা করতে পারে।

GrapheneOS-এ মূল MDM সক্ষমতা (Key MDM Capabilities on GrapheneOS):

- **রিমোট পলিসি এনফোর্সমেন্ট (Remote Policy Enforcement):** অ্যাডমিনিস্ট্রেটররা রিমোটলি পাসওয়ার্ড কমপ্লেক্সিটি রিকোয়ারমেন্ট (password complexity requirements) এনফোর্স করতে পারেন (যেমন- ন্যূনতম দৈর্ঘ্যের আলফানিউমেরিক পাসওয়ার্ড বাধ্যতামূলক করা), নির্দিষ্ট প্রোফাইলে ক্যামেরা অ্যাক্সেস ডিজেবল করতে পারেন, অ্যাপ্লিকেশন ইনস্টলেশনের সোর্স রেস্ট্রিক্ট (restrict) করতে পারেন এবং পরিচালিত সমস্ত ডিভাইসে ফোর্সফুলি VPN Kill Switch এনাবল করতে পারেন।
- **ওয়ার্ক প্রোফাইলস (Work Profiles):** Android Enterprise ডেডিকেটেড "Work Profile" তৈরি সাপোর্ট করে—যা হলো ইউজার প্রোফাইলের ভেতরে একটি ক্রিপ্টোগ্রাফিক্যালি আইসোলেটেড (cryptographically isolated) কন্টেইনার (container)। অ্যাডমিনিস্ট্রেটর Work Profile নিয়ন্ত্রণ করেন, সেখানে কর্পোরেট অ্যাপ্লিকেশন ইনস্টল করেন এবং কর্পোরেট পলিসি এনফোর্স করেন, আর অন্যদিকে ব্যবহারকারী তার পার্সোনাল প্রোফাইলের ওপর সম্পূর্ণ সার্বভৌমত্ব বা স্বনিয়ন্ত্রণ বজায় রাখেন। এই দুটি প্রোফাইলের মধ্যে ডেটা আদান-প্রদান করা অসম্ভব।
- **রিমোট ওয়াইপ (Remote Wipe):** ডিভাইস হারিয়ে গেলে বা চুরি হলে, অ্যাডমিনিস্ট্রেটর একটি রিমোট ওয়াইপ কমান্ড ইস্যু করতে পারেন যা একটি সম্পূর্ণ ফ্যাক্টরি রিসেট ট্রিগার করে এবং সমস্ত ইউজার ডেটা ধ্বংস করে দেয়। GrapheneOS-এ, এই কমান্ডটি Duress PIN (অধ্যায় ২.১৭)-এর মতোই Titan M2 কি ডেস্ট্রাকশন মেকানিজম (key destruction mechanism) ব্যবহার করে, যা ওয়াইপ প্রক্রিয়াটিকে ক্রিপ্টোগ্রাফিক এবং তাৎক্ষণিক (instantaneous) করে তোলে।
- **জিরো-টাচ এনরোলমেন্ট (Zero-Touch Enrollment):** লার্জ-স্কেল বা বড় পরিসরের ডিপ্লয়মেন্টের ক্ষেত্রে, ডিভাইসগুলোকে এমনভাবে প্রি-কনফিগার (pre-configured) করা যায় যেন প্রথমবার বুট করার সাথেই সেগুলো স্বয়ংক্রিয়ভাবে MDM সার্ভারে এনরোল হয়ে যায়। এর ফলে ম্যানুয়াল, পার-ডিভাইস সেটআপের (per-device setup) কোনো প্রয়োজন হয় পরিচয়।

রেকমেন্ডেড MDM প্ল্যাটফর্মস (Recommended MDM Platforms):

যেসব প্রতিষ্ঠান GrapheneOS ডিপ্লয় করতে চায়, তাদের কম্প্যাটিবিলিটি এবং প্রাইভেসি পোসচারের (privacy posture) ওপর ভিত্তি করে নিচের MDM সলিউশনগুলো রেকমেন্ড করা হয়:

- **Headwind MDM:** এটি একটি ওপেন-সোর্স, সেলফ-হোস্টেড (self-hosted) MDM সার্ভার। যেহেতু এটি আপনার নিজস্ব ইনফ্রাস্ট্রাকচারে রান করে, তাই কোনো থার্ড-পার্টি ক্লাউড প্রোভাইডারের কাছে ডিভাইসের কোনো টেলিমেট্রি (telemetry) ট্রান্সমিট করা হয় না। এক্সট্রিম ডেটা সোভেরেইনিটি বা চরম ডেটা সার্বভৌমত্বের রিকোয়ারমেন্ট থাকা প্রতিষ্ঠানগুলোর জন্য এটি একটি আদর্শ পছন্দ।
- **স্ট্যান্ডার্ড Android Enterprise MDMs:** যেসব কমার্শিয়াল প্ল্যাটফর্ম Android Enterprise স্ট্যান্ডার্ড ইমপ্লিমেন্ট করে (যেমন যারা Android Management API সাপোর্ট করে), সেগুলো সাধারণত GrapheneOS-এর সাথে কম্প্যাটিবল, যদি না তাদের ম্যানেজমেন্ট এজেন্টের (management agent) জন্য Google Play Services-এর

প্রয়োজন হয়। প্রতিষ্ঠানগুলোর উচিত GrapheneOS-এর Sandboxed Google Play কম্প্যাটিবিলিটি লেয়ারের সাথে এর কম্প্যাটিবিলিটি ভেরিফাই করা অথবা পুরোপুরি ডি-গুগলড প্রোফাইল দিয়ে টেস্ট করা।

ফ্লিট প্রোভিশনিং: দ্য অপারেশনাল প্রোটোকল (Fleet Provisioning: The Operational Protocol)

ফিল্ডে থাকা সাংবাদিকদের জন্য ৫০টি ডিভাইস ডিপ্লয় করতে চাওয়া কোনো নিউজ-রুম বা NGO-এর জন্য নিচের অপারেশনাল প্রোটোকলটি রেকমেন্ড করা হয়:

১. **হার্ডওয়্যার প্রকিউরমেন্ট (Hardware Procurement):** ক্লিন সাপ্লাই চেইন কাস্টডি (clean supply chain custody) নিশ্চিত করতে গুগলের অফিসিয়াল স্টোর থেকে সরাসরি একই মডেলের Pixel ডিভাইস ক্রয় করুন। থার্ড-পার্টি রিটেইলার এড়িয়ে চলুন যেখানে ডিভাইসগুলো আগে থেকেই কম্প্রোমাইজড থাকতে পারে। ২. **সেন্ট্রালাইজড ফ্ল্যাশিং (Centralized Flashing):** ওয়েব ইনস্টলার ব্যবহার করে (অধ্যায় ১.৪) সমস্ত ডিভাইসে GrapheneOS ফ্ল্যাশ করার জন্য একজন নির্ভরযোগ্য IT অ্যাডমিনিস্ট্রেটরকে দায়িত্ব দিন। সমস্ত বুটলোডার (bootloaders) লক করুন। প্রতিটি ডিভাইসে OEM Unlocking ডিজেবল করুন। ৩. **প্রোফাইল আর্কিটেকচার (Profile Architecture):** প্রতিটি ডিভাইসে User Profile blueprint (অধ্যায় ২.৬) প্রি-কনফিগার করুন। একটি স্টেরাইল বা জীবাণুমুক্ত (sterile) Owner profile, Signal এবং ProtonMail-সহ একটি সুরক্ষিত Daily profile এবং অর্গানাইজেশন-নির্দেশিত যেকোনো অ্যাপ্লিকেশনের জন্য Sandboxed Google Play-সহ একটি কোয়ারেন্টাইনড (quarantined) প্রোফাইল তৈরি করুন। ৪. **MDM এনরোলমেন্ট (MDM Enrollment):** প্রতিটি ডিভাইসকে অর্গানাইজেশনের MDM সার্ভারে এনরোল করুন। সিকিউরিটি বেইজলাইন পলিসি (security baseline policy) পুশ (push) করুন: VPN Kill Switch এনাবল করা, Private DNS কনফিগার করা, Auto-Reboot timer ৪ ঘণ্টায় সেট করা এবং USB-C Port Control এনফোর্স করা। ৫. **ডিস্ট্রিবিউশন (Distribution):** কর্মীদের মধ্যে ডিভাইসগুলো ডিস্ট্রিবিউট বা বিতরণ করার আগে একটি সংক্ষিপ্ত অপারেশনাল সিকিউরিটি ব্রিফিং (operational security briefing) দিন, যেখানে Reboot Rule (অধ্যায় ২.১৫) এবং প্রোফাইল সুইচিং প্রোটোকলের (profile switching protocol) বিষয়গুলো কভার করা হবে।

Seedvault-এর মাধ্যমে এনক্রিপ্টেড ব্যাকআপ (Encrypted Backups with Seedvault)

এন্টারপ্রাইজ ডিপ্লয়মেন্টের অন্যতম ক্রিটিক্যাল চ্যালেঞ্জ হলো Google Drive বা কোনো থার্ড-পার্টি ক্লাউড প্রোভাইডারের মাধ্যমে সংবেদনশীল ডেটা রাউট না করে নির্ভরযোগ্য ব্যাকআপ মেইনটেইন করা।

GrapheneOS একটি ফার্স্ট-ক্লাস (first-class) ব্যাকআপ সলিউশন হিসেবে **Seedvault**—একটি ওপেন-সোর্স, এনক্রিপ্টেড ব্যাকআপ ফ্রেমওয়ার্ক—ইন্টিগ্রেট করে।

Seedvault সিস্টেম লেভেলে অপারেট করে, যা অ্যাপ্লিকেশন ডেটা, ডিভাইসের সেটিংস এবং কল লগ ক্যাপচার করার জন্য অ্যান্ড্রয়েডের নেটিভ ব্যাকআপ API (native backup API) ব্যবহার করে। গুগলের ব্যাকআপ সার্ভিসের সাথে এর ক্রিটিক্যাল বা সবচেয়ে বড় পার্থক্য হলো, Seedvault সমস্ত ডেটাকে ডিভাইস থেকে বাইরে যাওয়ার আগেই লোকালি এনক্রিপ্ট (locally encrypt) করে। এই এনক্রিপশন কি-গুলো (encryption keys) একটি ইউজার-কন্ট্রোলড রিকভারি কোড (user-controlled recovery code) থেকে উদ্ভূত বা ডিরাইভড (derived) হয়—যা মূলত সেটআপের সময় জেনারেট করা একটি ১২ শব্দের নেমোনিক ফ্রেজ (12-word mnemonic phrase)।

ব্যাকআপ ডেস্টিনেশনস (Backup Destinations):

Seedvault একাধিক স্টোরেজ ব্যাকএন্ড (storage backends) সাপোর্ট করে, যার ফলে অর্গানাইজেশনগুলো তাদের ডেটা সোভেরেইনিটি রিকোয়ারমেন্টের সাথে সামঞ্জস্যপূর্ণ একটি ডেস্টিনেশন বেছে নিতে পারে:

- **USB Storage (Offline):** সবচেয়ে সুরক্ষিত অপশন। ব্যাকআপগুলো এনক্রিপ্ট করে সরাসরি ডিভাইসের USB-C পোর্টে যুক্ত করা একটি USB ফ্ল্যাশ ড্রাইভে (flash drive) রাইট করা হয়। এরপর ফ্ল্যাশ ড্রাইভটি কোনো ফিজিক্যাল সেইফ বা সিন্ধুকে (physical safe) নিরাপদে সংরক্ষণ করা যায়। এই পদ্ধতিটি সম্পূর্ণভাবে air-gapped—এখানে কোনো নেটওয়ার্ক কানেকশন জড়িত থাকে না।
- **Nextcloud (Self-Hosted Cloud):** যেসব প্রতিষ্ঠানের নেটওয়ার্ক-অ্যাক্সেসিবল (network-accessible) ব্যাকআপের প্রয়োজন হয়, তাদের জন্য Seedvault সরাসরি **Nextcloud**-এর সাথে ইন্টিগ্রেশন সাপোর্ট করে—যা হলো একটি ওপেন-সোর্স, সেলফ-হোস্টেড ক্লাউড প্ল্যাটফর্ম। ব্যাকআপগুলো ডিভাইসেই এনক্রিপ্ট করা হয় এবং তারপর WebDAV-এর মাধ্যমে অর্গানাইজেশনের ব্যক্তিগতভাবে পরিচালিত Nextcloud সার্ভারে আপলোড করা হয়। Nextcloud অ্যাডমিনিস্ট্রেটররাও এই ব্যাকআপ ডেটা ডিক্রিপ্ট করতে পারেন না, কারণ এনক্রিপশন কি-গুলো কখনোই ব্যবহারকারীর ডিভাইস ছেড়ে যায় না।
- **DAVx[5] ইন্টিগ্রেশন (DAVx[5] Integration):** অন্যান্য WebDAV-কম্প্যাটিবল ক্লাউড স্টোরেজ প্রোভাইডারদের জন্য, Seedvault DAVx[5] সিঙ্ক্রোনাইজেশন ক্লায়েন্টের মাধ্যমে এনক্রিপ্টেড ব্যাকআপ রাউট করতে পারে।

দ্য রিকভারি কোড প্রোটোকল (The Recovery Code Protocol):

Seedvault ইনিশিয়ালাইজ করার সময়, এটি একটি ১২-শব্দের BIP39-স্টাইল রিকভারি কোড (BIP39-style recovery code) জেনারেট করে। এই কোডটিই হলো সমস্ত ব্যাকআপ ডেটার ক্রিপ্টোগ্রাফিক কি। যদি ডিভাইসটি হারিয়ে যায়, চুরি হয়ে যায় বা ফ্যাক্টরি-ওয়াইপ (factory-wiped) করা হয়, তবে ব্যবহারকারী ১২-শব্দের কোডটি এন্টার করে একটি ফ্রেশ GrapheneOS ইনস্টলেশনের ওপর তাদের সম্পূর্ণ ব্যাকআপ রিস্টোর (restore) করতে পারেন।

> *[!WARNING]*
> ***দ্য ফরেনসিক Seedvault ভালনারেবিলিটি (The Forensic Seedvault Vulnerability):*** *DFRWS EU 2026 ফরেনসিক অ্যানালাইসিস পেপারে নিশ্চিত করা হয়েছে যে, যেহেতু Titan M2-এর ফিজিক্যাল এক্সপ্লয়টেশন (physical exploitation) কার্যত অসম্ভব, তাই একজন ইনভেস্টিগেটরের (investigator) জন্য ইউজার ডেটা অ্যাকোয়ার (acquire) করার সবচেয়ে প্রমিজিং বা সম্ভাবনাময় উপায় হলো একটি Seedvault ব্যাকআপ জব্দ করা। কোনো ইনভেস্টিগেটর যদি এনক্রিপ্টেড ব্যাকআপ থাকা একটি USB ড্রাইভ বা ক্লাউড অ্যাকাউন্ট জব্দ করেন এবং এর সাথে লিখে রাখা ১২-শব্দের রিকভারি ফ্রেজটিও আবিষ্কার করে ফেলেন, তবে ফোর্ট্রেসের (fortress) পতন ঘটে। তারা অনায়াসেই ব্যাকআপ রিস্টোর করে সমস্ত হার্ডওয়্যার এনক্রিপশন বাইপাস করে ফেলতে পারেন।*

এন্টারপ্রাইজ অ্যাডমিনিস্ট্রেটরদের অবশ্যই এই রিকভারি কোডগুলোর জন্য একটি কঠোর প্রোটোকল ইমপ্লিমেন্ট করতে হবে:

- প্রতিটি ব্যবহারকারীর রিকভারি কোড অবশ্যই ফিজিক্যাল কাগজে রেকর্ড করতে হবে (কখনোই ডিজিটালভাবে নয়) এবং একটি উচ্চমাত্রায় সুরক্ষিত, টেম্পার-এভিডেন্ট এনভেলপে (tamper-evident envelope) বা টেম্পার প্রুফ খামে সংরক্ষণ করতে হবে।
- সিল করা খামগুলো অবশ্যই এমন কোনো ফিজিক্যাল সেইফে জমা রাখতে হবে যেখানে কেবল IT অ্যাডমিনিস্ট্রেটর এবং ডিভাইসের মালিকের অ্যাক্সেস রয়েছে।
- রিকভারি কোডগুলো কখনই ইমেইল, মেসেজিং অ্যাপ বা কোনো ডিজিটাল চ্যানেলের মাধ্যমে ট্রান্সমিট বা প্রেরণ করা যাবে না।

রিমোট পলিসি এনফোর্সমেন্টের জন্য Android Enterprise MDM এবং এনক্রিপ্টেড, সেলফ-হোস্টেড ব্যাকআপের জন্য Seedvault-এর সমন্বয়ে অর্গানাইজেশনগুলো লার্জ স্কেলে GrapheneOS ডিপ্লয় করতে পারে, একই সাথে তাদের সম্পূর্ণ ডেটা সোভেরেইনিটি বা ডেটা সার্বভৌমত্ব বজায় রাখতে পারে। কোনো কর্পোরেট টেলিমেট্রি গুগলের কাছে পৌঁছায় না। কোনো থার্ড পার্টি কোনো ব্যাকআপ ডেটা রিড করতে পারে না। প্রোভিশনিং থেকে শুরু করে রিকভারি পর্যন্ত সম্পূর্ণ লাইফসাইকেল বা জীবনচক্রটি অর্গানাইজেশন নিজেই নিয়ন্ত্রণ করে—সার্ভিল্যান্স ইকোনমির (surveillance economy) কাছে এক বাইট ডেটাও সারেন্ডার বা সমর্পণ না করে।

ফ্লিট ডিপ্লয় এবং অপারেশনাল হয়ে যাওয়ার পর, একটি চূড়ান্ত ইনভিজিবল বা অদৃশ্য ইনফ্রাস্ট্রাকচার পুরো সিস্টেমটিকে সম্ভব করে তোলে: আর তা হলো আপডেট সার্ভারগুলো। পরিশিষ্ট গ (Appendix C)-তে, আমরা সেই সার্ভার আর্কিটেকচার নিয়ে পরীক্ষা করব যা একসাথে লাখ লাখ ডিভাইসে ক্রিপ্টোগ্রাফিক্যালি সাইনড ওএস আপডেট (cryptographically signed

OS updates) ডেলিভার করে—এবং সেই অ্যাডভান্সড ট্রাফিক ইঞ্জিনিয়ারিং (advanced traffic engineering) সম্পর্কে জানব যা লোডের নিচে ইনফ্রাস্ট্রাকচারটিকে ভেঙে পড়া থেকে রক্ষা করে।

পরিশিষ্ট গ: সার্ভার আর্কিটেকচার এবং CAKE ট্রাফিক শেপিং (Appendix C: Server Architecture & CAKE Traffic Shaping)

প্রতিটি GrapheneOS ডিভাইসের পেছনে রয়েছে এক অদৃশ্য ইনফ্রাস্ট্রাকচার বা অবকাঠামো: এমন একটি সার্ভার নেটওয়ার্ক (server network) যা বিশ্বব্যাপী লাখ লাখ ডিভাইসের জন্য অপারেটিং সিস্টেম আপডেটগুলো বিল্ড করে, সাইন করে, হোস্ট করে এবং ডিস্ট্রিবিউট বা বিতরণ করে। একজন সাধারণ এন্ড ইউজার (end user) কেবল একটি সাধারণ "System update available" নোটিফিকেশনের সাথেই ইন্টার‍্যাক্ট করেন, কিন্তু সেই নোটিফিকেশনের পেছনের ইঞ্জিনিয়ারিংটি হলো হাই-পারফরম্যান্স Linux systems administration-এর একটি মাস্টারক্লাস।

এই পরিশিষ্টে GrapheneOS আপডেট ডেলিভারি ইনফ্রাস্ট্রাকচারের একটি টেকনিক্যাল ওভারভিউ এবং সেই অ্যাডভান্সড নেটওয়ার্ক ট্রাফিক ম্যানেজমেন্ট (network traffic management) টেকনিকগুলো নিয়ে আলোচনা করা হয়েছে, যা বিশাল স্কেলে অত্যন্ত নির্ভরযোগ্য, ন্যায্য এবং হাই-স্পিড আপডেট ডিস্ট্রিবিউশন নিশ্চিত করে।

আপডেট ডেলিভারি আর্কিটেকচার (The Update Delivery Architecture)

GrapheneOS স্ট্যাটিক ফাইল হোস্ট (static file hosts) হিসেবে পরিচালিত ডেডিকেটেড সার্ভারের একটি নেটওয়ার্কের মাধ্যমে over-the-air (OTA) আপডেটগুলো ডিস্ট্রিবিউট করে। আর্কিটেকচারটি উদ্দেশ্যপ্রণোদিতভাবেই অত্যন্ত সরল এবং শক্তিশালী রাখা হয়েছে। আপডেট রিকোয়েস্ট হ্যান্ডেল করার জন্য এখানে কোনো জটিল অ্যাপ্লিকেশন সার্ভার নেই, কোনো ডায়নামিক API লেয়ার নেই এবং কোনো ডেটাবেস ব্যাকএন্ডও (database backend) নেই। আপডেট ফাইলগুলো—ফুল OTA প্যাকেজ এবং ইনক্রিমেন্টাল ডেল্টা (incremental delta) আপডেট উভয়ই—হলো আগে থেকে সাইন করা (pre-signed), আগে থেকে জেনারেট করা স্ট্যাটিক ফাইল, যা সরাসরি HTTPS-এর মাধ্যমে সার্ভ করা হয়।

প্রতিটি ডিভাইসে থাকা GrapheneOS Updater client নির্দিষ্ট সময় পরপর আপডেট সার্ভারের channel metadata file (একটি লাইটওয়েট টেক্সট ফাইল যাতে লেটেস্ট ভার্সন নম্বর, ফাইলের সাইজ এবং cryptographic hash থাকে) পোল (poll) করে চেক করে। যদি নতুন কোনো ভার্সন ডিটেক্ট বা শনাক্ত হয়, তবে ক্লায়েন্ট সরাসরি স্ট্যাটিক ফাইল সার্ভার থেকে আপডেট প্যাকেজটি ডাউনলোড করে, লোকালি এর cryptographic signature ভেরিফাই করে এবং ইনস্টলেশনটি স্টেজ (stage) করে।

এই স্ট্যাটিক আর্কিটেকচারটি বেশ কিছু ক্রিটিক্যাল বা অত্যন্ত গুরুত্বপূর্ণ সুবিধা প্রদান করে:

- **ন্যূনতম অ্যাটাক সারফেস (Minimal Attack Surface):** এক্সপ্লয়েট করার মতো কোনো ডায়নামিক ব্যাকএন্ড এখানে নেই। সার্ভারটি কেবল TLS-এর ওপর দিয়ে ফাইল সার্ভ করে। এখানে কোনো SQL injection ভেক্টর নেই,

কোনো অথেনটিকেশন API নেই এবং কোনো সেশন ম্যানেজমেন্ট ভালনারেবিলিটিও নেই।

- **ক্যাশেবিলিটি (Cacheability):** স্ট্যাটিক ফাইলগুলো স্বভাবতই ক্যাশেবল (cacheable), যা Content Delivery Network (CDN) এজ নোডগুলোকে (edge nodes) মূল সার্ভারগুলোর ওপর কোনো চাপ না ফেলেই অধিকাংশ ডাউনলোড ট্রাফিক অ্যাবজর্ব (absorb) করে নিতে সহায়তা করে।
- **রিপ্রোডিউসিবিলিটি (Reproducibility):** যেহেতু আপডেট ফাইলগুলো আগে থেকেই একটি air-gapped বিল্ড ইনফ্রাস্ট্রাকচারে জেনারেট এবং সাইন করা থাকে, তাই সার্ভিং লেয়ারটি কখনই কোনো সংবেদনশীল ক্রিপ্টোগ্রাফিক মেটেরিয়াল স্পর্শ বা প্রসেস করে না।

দ্য বাফারব্লোট প্রবলেম (The Bufferbloat Problem)

যখন GrapheneOS-এর কোনো বড় সিকিউরিটি আপডেট রিলিজ হয়, তখন বিশ্বব্যাপী লাখ লাখ ডিভাইস একই সাথে মাল্টি-গিগাবাইট OTA প্যাকেজ ডাউনলোড করা শুরু করে। এটি আপডেট ইনফ্রাস্ট্রাকচারের বিরুদ্ধে কনকারেন্ট (concurrent) বা সমসাময়িক TCP connections-এর একটি বিশাল ও আকস্মিক সার্জ বা ঢেউ তৈরি করে।

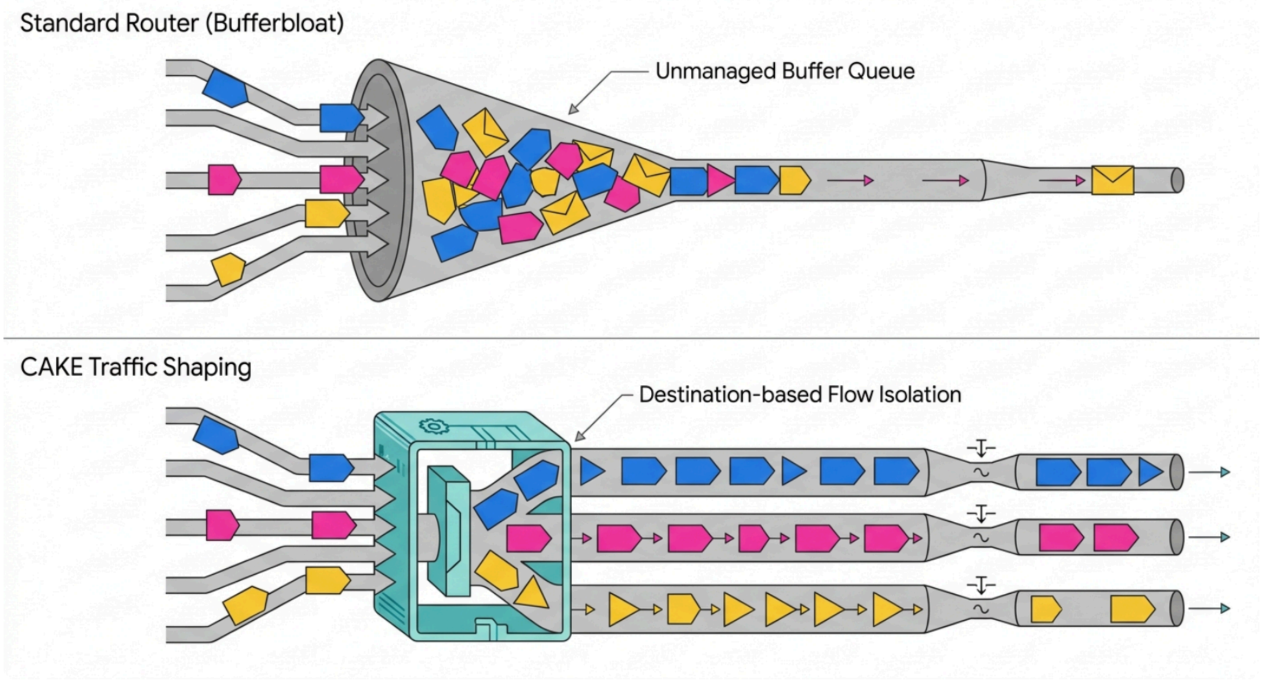
একটি সাধারণভাবে কনফিগার করা সার্ভারে, এই ট্রাফিকের আকস্মিক ঢেউটি **Bufferbloat (বাফারব্লোট)** নামক একটি নেটওয়ার্কিং ফেনোমেনন বা ঘটনার জন্ম দেয়।

বাফারব্লোট তখন ঘটে, যখন নেটওয়ার্ক ইন্টারফেসের transmit queue (যে বাফারে আউটগোয়িং বা বহির্গামী প্যাকেটগুলো নেটে পাঠানোর আগে অপেক্ষা করে) অত্যাধিক মাত্রায় বড় হয়ে যায়। অপারেটিং সিস্টেমের ডিফল্ট TCP স্ট্যাক ক্রমাগত এই বাফারে প্যাকেট ফিড (feed) করতে থাকে এবং এই বিপুল পরিমাণ প্যাকেট সামাল দিতে গিয়ে বাফারটি ফুলেফেঁপে ওঠে। যদিও তাত্ত্বিকভাবে কোনো প্যাকেট "ড্রপ" (dropped) হয় না, কিন্তু প্রতিটি প্যাকেটের ল্যাটেন্সি (latency) বা বিলম্ব নাটকীয়ভাবে বৃদ্ধি পায়—কয়েক মিলি-সেকেন্ড থেকে তা কয়েক সেকেন্ডে গিয়ে দাঁড়ায়।

এর ফলস্বরূপ ইউজার এক্সপেরিয়েন্স চরম মাত্রায় ক্ষতিগ্রস্ত হয়। ডাউনলোডের স্পিড একদম কমে যায় বা কলাপ্স (collapse) করে। TCP connections টাইম-আউট হয়ে যায় এবং একদম শুরু থেকে রিস্টার্ট (restart) নেয়। যেসব ডিভাইসের ডাউনলোড ৯০% সম্পন্ন হয়ে গিয়েছিল, সেগুলোও বাধ্য হয় আবার নতুন করে শুরু করতে। যদিও সার্ভারের CPU এবং মেমরি খুব কমই ব্যবহৃত হয়, তবুও সার্ভারটিকে ওভারলোডেড বা লোড নিতে অক্ষম বলে মনে হয়—কারণ বটলনেক (bottleneck) বা মূল প্রতিবন্ধকতাটি পুরোপুরি তৈরি হয় ওই ফুলে ওঠা নেটওয়ার্ক কিউয়ের (network queue) ভেতরেই।

দ্য CAKE সলিউশন (The CAKE Solution)

Bufferbloat Mitigation: Standard FIFO vs. CAKE Queue Management



By intentionally restricting outbound bandwidth slightly below the physical link capacity, CAKE ensures the server—not an external, unmanaged router—retains control over packet queuing, utilizing **dual-dsthost** isolation to guarantee fair bandwidth distribution per client.

বাফারব্লোট সম্পূর্ণ নির্মূল করতে এবং ন্যায্য ও হাই-পারফরম্যান্স আপডেট ডিস্ট্রিবিউশন নিশ্চিত করতে, GrapheneOS ইনফ্রাস্ট্রাকচার **CAKE (Common Applications Kept Enhanced)** ডিপ্লয় করে—যা হলো একটি অ্যাডভান্সড, Linux kernel-লেভেলের ট্রাফিক শেপিং (traffic shaping) ডিসিপ্লিন।

CAKE হলো Linux kernel-এর `tc` (traffic control) সাবসিস্টেমের অংশ হিসেবে ডেভেলপ করা একটি অত্যন্ত অত্যাধুনিক কিউ ম্যানেজমেন্ট সিস্টেম (queue management system)। এটি ডিফল্ট FIFO (First-In, First-Out) transmit queue-কে একটি ইন্টেলিজেন্ট, মাল্টি-লেয়ার্ড শিডিউলিং অ্যালগরিদম (multi-layered scheduling algorithm) দিয়ে রিপ্লেস করে বা প্রতিস্থাপন করে।

CAKE যেভাবে কাজ করে (How CAKE operates):

১. অ্যাক্টিভ কিউ ম্যানেজমেন্ট (Active Queue Management - AQM): CAKE একটি মডিফায়েড COBALT AQM অ্যালগরিদম (Codel এবং BLUE-এর সমন্বয়) ইন্টিগ্রেট করে। ট্রান্সমিট বাফারটিকে অনির্দিষ্টকালের জন্য ফুলে উঠতে দেওয়ার পরিবর্তে, COBALT অত্যন্ত স্ট্র্যাটেজিক বা কৌশলগতভাবে প্যাকেটগুলোকে ড্রপ করে অথবা সেগুলোতে ECN-marks যুক্ত করে, এবং এই কাজটি কিউটি বাফারব্লোটেড হওয়ার *আগেই* করা হয়। কিউয়ের ডেপথ বা গভীরতা প্রোঅ্যাক্টিভলি (proactively) ম্যানেজ করার মাধ্যমে, CAKE চরম কনকারেন্ট লোডের (concurrent load) মধ্যেও ধারাবাহিকভাবে ল্যাটেন্সিকে অত্যন্ত নিচু পর্যায়ে ধরে রাখে।

২. ফ্লো ফেয়ারনেস (Flow Fairness - FQ): CAKE পার-ফ্লো ফেয়ার কিউইং (per-flow fair queuing) ইমপ্লিমেন্ট করে। যখন ১,০০,০০০ ডিভাইস একই সাথে একটি আপডেট ডাউনলোড করে, তখন CAKE সম্পূর্ণ ট্রাফিকটিকে একটি একক, মনোলিথিক স্ট্রিম (monolithic stream) হিসেবে বিবেচনা করে না। এটি প্রতিটি আলাদা TCP connection-কে (অর্থাৎ প্রতিটি ডিভাইসের ডাউনলোডকে) একটি পৃথক "ফ্লো" (flow) হিসেবে শনাক্ত করে এবং সমস্ত অ্যাক্টিভ ফ্লোয়ের মধ্যে সমানভাবে ব্যান্ডউইথ (bandwidth) বণ্টন করে দেয়।

এটি একটি একক ও অ্যাগ্রেসিভ কানেকশনকে পুরো লিংকটি একচেটিয়াভাবে দখল করতে বাধা দেয়। ফাইবার অপটিক (fiber optic) ব্যবহার করে একটি ডিভাইস সর্বোচ্চ গতিতে ডাউনলোড করুক বা ধীর গতির সেলুলার কানেকশন ব্যবহার করে অন্য একটি ডিভাইস অল্প অল্প করে ডেটা টানুক, CAKE এটি নিশ্চিত করে যে প্রতিটি ফ্লো-ই উপলব্ধ ব্যান্ডউইথের তার ন্যায্য অংশটুকু পাবে।

৩. ব্যান্ডউইথ শেপিং (Bandwidth Shaping): CAKE অ্যাডমিনিস্ট্রেটরদের এমন একটি হার্ড ব্যান্ডউইথ সিলিং (hard bandwidth ceiling) সেট করার অনুমতি দেয়, যা সার্ভারের ফিজিক্যাল লিংক ক্যাপাসিটির (physical link capacity) চেয়ে সামান্য কম থাকে। কনফিগার করা রেটটিকে ইচ্ছাকৃতভাবে আন্ডার-প্রোভিশনিং (under-provisioning) করার মাধ্যমে (যেমন- 1 Gbps লিংকে CAKE-কে 940 Mbps-এ সেট করা), ট্রাফিক শেপিংটি আপস্ট্রিম (upstream) রাউটারের নির্বোধ FIFO বাফারের পরিবর্তে সার্ভারের নিজস্ব ইন্টেলিজেন্ট CAKE কিউয়ের ভেতরেই সম্পন্ন হয়। এটি অত্যন্ত গুরুত্বপূর্ণ কারণ সার্ভারের CAKE কিউটি ইন্টেলিজেন্ট (যা AQM এবং fairness প্রয়োগ করে), কিন্তু ISP-এর আপস্ট্রিম বাফারটি তা নয়।

৪. ডিফসর্ভ অ্যাওয়ারনেস (Diffserv Awareness): CAKE আইপি প্যাকেট হেডারে (IP packet headers) Differentiated Services (Diffserv) মার্কিংগুলো শনাক্ত করতে পারে। বিশাল, থ্রুপুট-ওরিয়েন্টেড (throughput-oriented) OTA ডাউনলোড স্ট্রিমগুলোর ওপর লাইটওয়েট মেটাডেটা পোলিং রিকোয়েস্টগুলোকে (যেগুলো মূলত খুব ছোট এবং ল্যাটেন্সি-সেনসিটিভ HTTP ফেচ) প্রাইওরিটাইজ (prioritize) বা অগ্রাধিকার দেওয়া যায়। এটি নিশ্চিত করে যে, আপডেটের জন্য চেক করা ডিভাইসগুলো ডাউনলোডের পিক সার্জের (peak download surges) সময়ও ইনস্ট্যান্ট রেসপন্স বা তাৎক্ষণিক সাড়া পায়।

দ্য অপারেশনাল ইফেক্ট (The Operational Effect)

আপডেট সার্ভারগুলোতে CAKE ডিপ্লয় করার ফলে, GrapheneOS ইনফ্রাস্ট্রাকচার বেশ কয়েকটি ক্রিটিক্যাল পারফরম্যান্স গ্যারান্টি অর্জন করে:

- পিক আপডেট ডিস্ট্রিবিউশন ইভেন্টের সময়ও **সাব-মিলিসেকেন্ড কিউ ল্যাটেন্সি (Sub-millisecond queue latency)** বজায় থাকে।
- **ফেয়ার পার-ডিভাইস ব্যান্ডউইথ অ্যালোকেশন (Fair per-device bandwidth allocation)** বা ডিভাইসভিত্তিক ন্যায্য ব্যান্ডউইথ বণ্টন নিশ্চিত হয়, যা ফাস্ট কানেকশনগুলোকে স্লো কানেকশনগুলোকে বঞ্চিত করা থেকে বিরত রাখে।
- TCP টাইমআউট-প্ররোচিত ডাউনলোড রিস্টার্টের পরিমাণ **একেবারে জিরো (Zero TCP timeout-induced download restarts)**-তে নেমে আসে, কারণ ল্যাটেন্সি ধারাবাহিকভাবে এত কম থাকে যা সমস্ত কানেকশনকে অ্যাক্টিভ রাখতে সক্ষম।
- বাফারব্লোটেড সার্ভারগুলোর মতো ক্যাটাস্ট্রফিক কলাপ্স (catastrophic collapse) বা চরম বিপর্যয়ের বদলে এক্সট্রিম লোডের (extreme load) নিচে অত্যন্ত **গ্রেসফুল ডিগ্রেডেশন (Graceful degradation)** ঘটে।

এর ফলস্বরূপ, লাখ লাখ GrapheneOS ডিভাইস একই সাথে ইনফ্রাস্ট্রাকচারকে ওভারহোয়েমড (overwhelmed) না করেই মাল্টি-গিগাবাইট সিকিউরিটি আপডেটগুলো নির্বিঘ্নে পুল (pull) বা ডাউনলোড করতে পারে। এর জন্য সার্ভারটিকে হার্ডওয়্যারের দিক থেকে ব্যাপকভাবে ওভার-প্রোভিশন্ড (over-provisioned) করার কোনো প্রয়োজন হয় না। CAKE এটি নিশ্চিত করে যে উপলব্ধ ব্যান্ডউইথটি অত্যন্ত অপটিমালি বা সর্বোত্তমভাবে, ন্যায্যভাবে এবং কোনো ল্যাটেন্সি স্পাইক (latency spikes) ছাড়াই ব্যবহৃত হচ্ছে, যা সাধারণত ডাউনলোডের রিলায়েবিলিটি বা নির্ভরযোগ্যতা ধ্বংস করে দেয়।

ইনফ্রাস্ট্রাকচার সিকিউরিটি যদিও অদৃশ্য, কিন্তু এটিই হলো সেই বেডরক (bedrock) বা ভিত্তিপ্রস্তর যার ওপর ভিত্তি করে সম্পূর্ণ আপডেট ডেলিভারি চেইনটি শক্তভাবে দাঁড়িয়ে আছে।

আপনার ডিভাইসের সিলিকনে ফিউজ হওয়া Titan M2 থেকে শুরু করে পৃথিবীর অপর প্রান্তে থাকা আপডেট সার্ভারগুলোর মধ্য দিয়ে প্রবাহিত হওয়া CAKE-শেপড (CAKE-shaped) ট্রাফিক পর্যন্ত—GrapheneOS ইকোসিস্টেমের প্রতিটি লেয়ার বা স্তর একটিমাত্র আপসহীন ম্যান্ডেট (mandate) নিয়ে তৈরি করা হয়েছে: আর তা হলো, আপনার সিকিউরিটি নন-নেগোসিয়েবল (non-negotiable)। এই বইটির পরিসমাপ্তিতে এসে, আমরা সেই ম্যান্ডেটটি ঠিক কী অর্থ বহন করে তা নিয়ে ভাবতে চাই—শুধু প্রযুক্তির জন্যই নয়, বরং স্বয়ং মানুষের প্রাইভেসি বা গোপনীয়তার ভবিষ্যতের জন্যও।

পরিশিষ্ট ঘ: মাস্টার রেফারেন্স বিবলিওগ্রাফি (Appendix D: Master Reference Bibliography)

এই গাইডটি গবেষণা এবং রচনার ক্ষেত্রে নিচের টেকনিক্যাল ডকুমেন্টেশন, ভালনারেবিলিটি রিপোর্ট এবং প্রাইমারি সোর্সগুলো (primary sources) ব্যবহার করা হয়েছে।

GrapheneOS মেমরি সেইফটি রিসার্চ (GrapheneOS Memory Safety Research)

- https://www.coalitioninc.com/au/blog/cyber-insurance/after-mythos-what-actually-changes-for-cyber-risk
- https://www.crowdstrike.com/en-us/blog/analysis-cve-2013-3906-exploit/
- https://vertu.com/lifestyle/ultimate-android-security-ranking-android-16-vs-grapheneos-titan-m2
- https://developer.arm.com/community/arm-community-blogs/b/architectures-and-processors-blog/posts/enhanced-security-through-mte
- https://ir0nstone.gitbook.io/notes/binexp/stack/memory-tagging-extension-mte
- https://www.linaro.org/blog/type-tracking-using-arm-memory-tagging/
- https://v8.dev/blog/retrofitting-temporal-memory-safety-on-c++
- https://www.reddit.com/r/PrivatePackets/comments/1qt212j/the_security_gap_between_grapheneos_and_standard/
- https://www.cape.co/blog/grapheneos-vs-android
- https://llvm.org/docs/ScudoHardenedAllocator.html
- https://releases.llvm.org/10.0.0/docs/ScudoHardenedAllocator.html
- https://grapheneos.social/@GrapheneOS/115249202253174791
- https://www.usenix.org/conference/woot24/presentation/mao

- https://synacktiv.com/en/publications/exploring-grapheneos-secure-allocator-hardened-malloc
- https://grapheneos.org/features
- https://github.com/GrapheneOS/hardened_malloc
- https://grapheneos.org/usage
- https://clehaxze.tw/gemlog/2022/04-30-security-web-applications-beyond-just-containers.gmi
- https://grapheneos.org/faq
- https://discuss.grapheneos.org/d/8439-mte-support-status-for-grapheneos
- https://developer.android.com/ndk/guides/arm-mte
- https://grapheneos.social/@GrapheneOS/115177759496849433
- https://developer.arm.com/-/media/Arm%20Developer%20Community/PDF/Arm_Memory_Tagging_Extension_Whitepaper.pdf
- https://codesandbox.io/p/github/aldorithms/hardened_malloc_explained
- https://www.usenix.org/system/files/woot24-mao.pdf
- https://cwe.mitre.org/data/definitions/416.html
- https://faculty.cc.gatech.edu/~milos/kharbutli_asplos06.pdf
- https://security.apple.com/blog/towards-the-next-generation-of-xnu-memory-safety/
- https://www.l3harris.com/newsroom/editorial/2023/10/scudo-hardened-allocator-unofficial-internals-documentation
- https://rocm.docs.amd.com/projects/llvm-project/en/latest/LLVM/llvm/html/ScudoHardenedAllocator.html
- https://textbook.cs161.org/memory-safety/mitigations.html
- https://obscurix.github.io/security/hardened-malloc.html
- https://ciq.com/blog/how-the-hardened-malloc-library-protects-processes-from-security-exploitation-on-rocky-linux-from-ciq-hardened-rlc-h/
- https://developer.arm.com/documentation/108035/0100/Introduction-to-the-Memory-Tagging-Extension
- https://www.darknavy.org/blog/strengthening_the_shield_mte_in_memory_allocators/
- https://developer.arm.com/documentation/108035/0100/How-does-MTE-work-
- https://news.ycombinator.com/item?id=39669487

- https://www.allpcb.com/allelectrohub/armv85-a-memory-tagging-extension-mte-explained
- https://discuss.grapheneos.org/d/8439-mte-support-status-for-grapheneos?page=2
- https://hacktricks.wiki/en/binary-exploitation/common-binary-protections-and-bypasses/memory-tagging-extension-mte.html
- https://source.android.com/docs/security/test/memory-safety/arm-mte
- https://discuss.grapheneos.org/d/11196-hardened-malloc-functionality
- https://8ksec.io/arm64-reversing-and-exploitation-part-10-intro-to-arm-memory-tagging-extension-mte/
- https://developer.arm.com/documentation/108035/0100/How-does-MTE-work-/MTE-modes
- https://arxiv.org/html/2509.22027v1
- https://grapheneos.org/releases
- https://discuss.grapheneos.org/d/7615-understanding-grapheneos-sandbox-and-permissions-in-vulnerability-scenarios
- https://www.usenix.org/event/woot10/tech/full_papers/Vanegue.pdf
- https://www.reddit.com/r/GrapheneOS/comments/16ikhlj/does_graphene_protect_against_0_day/

GrapheneOS অ্যাপ স্পনিং এক্সপ্লেইনড (GrapheneOS App Spawning Explained)

- https://synacktiv.com/en/publications/exploring-grapheneos-secure-allocator-hardened-malloc
- https://www.longlu.org/publication/lee-2014/
- https://grapheneos.org/usage
- https://wenke.gtisc.gatech.edu/papers/morula.pdf
- https://github.com/GrapheneOS/os-issue-tracker/issues/2170
- https://www.usenix.org/publications/loginonline/choose-one-android-performance-or-security
- https://man.exherbo.org/syd.7.html
- https://source.android.com/docs/core/runtime/zygote

- https://infosecwriteups.com/exploiting-android-zygote-injection-cve-2024-31317-d83f69265088
- https://grapheneos.org/releases
- https://copperhead.co/blog/aslr-android-zygote/
- https://www.reddit.com/r/degoogle/comments/u7einf/has_the_performance_on_grapheneos_improved_since/
- https://discuss.grapheneos.org/d/2733-does-using-grapheneos-make-you-stand-out
- https://netsec.ccert.edu.cn/files/papers/icc14-android.pdf
- https://discuss.grapheneos.org/u/spring-onion/likes
- https://rtx.meta.security/exploitation/2024/06/03/Android-Zygote-injection.html
- https://grapheneos.org/features#closed-device-identifier-leaks
- https://github.com/iAnonymous3000/awesome-grapheneos-guide
- https://grapheneos.org/features
- https://www.reddit.com/r/degoogle/comments/1rnhd36/how_many_of_you_actually_run_grapheneos_is_it/
- https://eylenburg.github.io/android_comparison.htm
- https://www.reddit.com/r/GrapheneOS/comments/17lty85/ram_management_issues/
- https://discuss.grapheneos.org/d/11561-does-grapheneos-run-faster-on-older-devices
- https://discuss.grapheneos.org/d/15476-allowlist-apps-to-ignore-secure-app-spawning
- https://discuss.grapheneos.org/d/33687-pixel-10a
- https://discuss.grapheneos.org/d/6158-consequences-of-disabling-secure-app-spawning
- https://www.reddit.com/r/GrapheneOS/comments/1ozuzid/hows_the_ram_usage/
- https://discuss.grapheneos.org/d/1636-general-question-about-the-state-of-graphene-and-overall-performance
- https://discuss.grapheneos.org/d/31269-grapheneos-version-2026012800-released
- https://discuss.grapheneos.org/d/8330-app-compatibility-with-grapheneos?page=3
- https://discuss.grapheneos.org/d/21036-xtwitter-now-requires-google-play/38
- https://discuss.grapheneos.org/d/21036-xtwitter-now-requires-google-play/33

GrapheneOS সিকিউরিটি রিসার্চ এবং চেকলিস্ট (GrapheneOS Security Research and Checklist)

- https://grapheneos.social/@GrapheneOS/112826164564944187
- https://riscv.org/blog/what-is-the-titan-m2-security-chip-in-googles-pixel-phones-calvin-wankhede-android-authority/
- https://android-developers.googleblog.com/2018/10/building-titan-better-security-through.html
- https://kashifmukhtar.com/pixel-10-titan-m2-security-hardware-root-trust/
- https://attestation.app/about
- https://www.reddit.com/r/GrapheneOS/comments/1o3fl5m/what_stops_attackers_from_brute_forcing_a_4_digit/
- https://discuss.grapheneos.org/d/12848-claims-made-by-forensics-companies-their-capabilities-and-how-grapheneos-fares/139
- https://www.bsidestriad.org/research/most-secure-mobile-os
- https://wccftech.com/google-explains-how-the-titan-m-chip-helps-secure-the-pixel-3/
- https://discuss.grapheneos.org/d/5136-brute-force-resistance-noob-question
- https://discuss.grapheneos.org/d/13107-grapheneos-and-forensic-extraction-of-data
- https://discuss.grapheneos.org/d/24356-cold-brute-force-attack
- https://grapheneos.social/@GrapheneOS/114721784437281675
- https://discuss.grapheneos.org/d/12848-claims-made-by-forensics-companies-their-capabilities-and-how-grapheneos-fares/62
- https://news.ycombinator.com/item?id=46038241
- https://grapheneos.org/faq
- https://discuss.grapheneos.org/d/22612-attestation-wrong-ways-what-not-to-do-what-to-expect
- https://grapheneos.org/articles/attestation-compatibility-guide
- https://discuss.grapheneos.org/d/8235-can-someone-explain-to-me-the-use-case-for-the-auditor-app-with-avb-enabled
- https://attestation.app/tutorial
- https://github.com/GrapheneOS/AttestationServer
- https://discuss.grapheneos.org/d/11799-remote-attestation-mechanics

- https://discuss.grapheneos.org/d/31616-pc-security-against-physical-attacks-introduce-me
- https://news.ycombinator.com/item?id=31167541
- https://9to5google.com/2018/10/17/titan-m-pixel-3-security/
- https://discuss.grapheneos.org/d/26248-leaving-bootloader-unlocked
- https://discuss.grapheneos.org/d/26248-leaving-bootloader-unlocked?page=2
- https://grapheneos.org/features
- https://discuss.grapheneos.org/d/10462-what-information-can-forensic-tools-recover-from-an-afu-gos-phone
- https://discuss.grapheneos.org/d/7290-how-to-protect-phone-from-content-extraction
- https://discuss.grapheneos.org/d/13155-grapheneos-version-2024053100-released/41
- https://ventral.digital/posts/2024/12/9/practical-grapheneos-for-the-paranoid/
- https://www.reddit.com/r/GrapheneOS/comments/1s0rmta/privacybackdoors_on_google_hardware_through/

GrapheneOS স্যান্ডবক্সড প্লে রিসার্চ (GrapheneOS Sandboxed Play Research)

- https://www.reddit.com/r/GrapheneOS/comments/12a3r32/graphenes_sandboxed_play_services_vs_microg_on/
- https://grapheneos.org/features#sandboxed-google-play
- https://discuss.grapheneos.org/d/29453-help-me-understand-this-google-play-services-being-sandboxed
- https://discuss.grapheneos.org/d/778-understanding-sandboxed-google-vs-microg
- https://grapheneos.org/
- https://source.android.com/docs/core/permissions/perms-allowlist
- https://medium.com/@sreelakshmis.dilip/android-automotive-understanding-app-permissions-and-privileged-applications-c00b83df11d6
- https://grapheneos.org/usage#android-auto
- https://grapheneos.org/usage
- https://discuss.grapheneos.org/d/10317-mysterious-gmscompat-message

- https://www.reddit.com/r/degoogle/comments/1oglb18/can_anyone_explain_what_is_sandboxed_google_play/
- https://grapheneos.social/@GrapheneOS/113459782313987260
- https://discuss.grapheneos.org/d/2155-gmscompatconfig-and-gmscompat
- https://grapheneos.org/releases
- https://github.com/GrapheneOS/platform_frameworks_opt_telephony
- https://grapheneos.org/features
- https://github.com/GrapheneOS/os-issue-tracker/issues/7352
- https://discuss.grapheneos.org/d/29571-new-gms-location-rerouting
- https://discuss.grapheneos.org/d/29020-can-someone-explain-reroute-location-requests-to-the-os
- https://grapheneos.org/usage#sandboxed-google-play
- https://en.wikipedia.org/wiki/GrapheneOS
- https://www.reddit.com/r/GrapheneOS/comments/14g0obf/limitations_of_graphenos/
- https://grapheneos.org/articles/attestation-compatibility-guide
- https://discuss.grapheneos.org/d/475-wallet-google-pay?page=7
- https://www.ford.com/support/how-tos/sync/getting-started-with-sync/how-does-android-auto-work-with-my-vehicle/
- https://www.reddit.com/r/LinusTechTips/comments/1g46pys/who_knew_that_wireless_capability_was_region/
- https://discuss.grapheneos.org/d/5403-newly-installed-pixel-6-wi-fi-keeps-disconnecting?page=2
- https://discuss.grapheneos.org/d/5403-newly-installed-pixel-6-wi-fi-keeps-disconnecting/28
- https://discuss.grapheneos.org/d/14014-android-auto-not-working
- https://discuss.grapheneos.org/d/15586-android-auto-working-but-are-these-expected-limitations
- https://discuss.grapheneos.org/d/10793-clarification-about-microg-what-is-it-is-it-insecure
- https://eylenburg.github.io/android_comparison.htm
- https://discuss.grapheneos.org/d/4290-sandboxed-microg

- https://news.ycombinator.com/item?id=28091157
- https://discuss.grapheneos.org/d/12056-fido2-security-keys-on-grapheneos-a-summary/44
- https://discuss.grapheneos.org/d/12056-fido2-security-keys-on-grapheneos-a-summary/55
- https://www.privacyguides.org/articles/2022/04/21/grapheneos-or-calyxos/

GrapheneOS অপসেক নেটওয়ার্ক ডিফেন্সেস (GrapheneOS OpSec Network Defenses)

- https://discuss.grapheneos.org/d/16142-note-you-may-want-to-turn-off-full-mac-randomization
- https://discuss.grapheneos.org/d/16142-note-you-may-want-to-turn-off-full-mac-randomization?page=2
- https://grapheneos.org/features
- https://github.com/GrapheneOS/os-issue-tracker/issues/3441
- https://discuss.grapheneos.org/d/7154-understanding-mac-addresses
- https://grapheneos.org/usage
- https://discuss.grapheneos.org/d/16593-wlan-settings-send-device-name
- https://grapheneos.org/faq
- https://discuss.grapheneos.org/d/1851-lte-only-mode-threat-model
- https://discuss.grapheneos.org/d/3942-baseband-vulnerabilities
- https://zimperium.com/glossary/imsi-catcher
- https://www.usenix.org/system/files/conference/woot17/woot17-paper-park.pdf
- https://palindrometech.com/applied-security-research-blog/an-evolutionary-analysis-of-cellular-network-security-vulnerabilities-and-protections-from-2g-to-5g
- https://sls.eff.org/technologies/cell-site-simulators-imsi-catchers
- https://godarkbags.com/blogs/post/imsi-catchers
- https://discuss.grapheneos.org/d/34257-trying-to-understand-dns-vpn-and-leak-issues
- https://discuss.grapheneos.org/d/14283-dns-leak
- https://discuss.grapheneos.org/d/10347-vpn-leaks-without-default-enabled-toggle-for-blocking-leaks-enabled

- https://discuss.grapheneos.org/d/16161-grapheneos-fixing-the-standard-vpn-leak-blocking-is-nearing-completion
- https://grapheneos.social/@GrapheneOS/113234908700821366
- https://medium.com/@nurkholish.halim/a-deep-dive-into-ebpf-writing-an-efficient-dns-monitoring-2c9dea92abdf
- https://discuss.grapheneos.org/d/10650-drm-provisioning-and-internet-access-pinning-why-choose-gos-servers
- https://discuss.grapheneos.org/d/14818-dns-leaks
- https://discuss.grapheneos.org/d/28682-maximum-security-and-anonymity
- https://discuss.grapheneos.org/d/23736-automatic-18-hour-reboots
- https://sideofburritos.com/blog/pin-code-thefts/
- https://discuss.grapheneos.org/d/5069-auto-reboot-security-issue
- https://discuss.grapheneos.org/d/33331-rethinkdns-ip-leaks/18

GrapheneOS অ্যাপ পারমিশনস এক্সপ্লেইনড (GrapheneOS App Permissions Explained)

- https://en.wikipedia.org/wiki/GrapheneOS
- https://www.youtube.com/watch?v=wMIALUERw3g
- https://www.reddit.com/r/degoogle/comments/vyr2xw/grapheneos_with_new_killer_features_storage/
- https://grapheneos.org/features
- https://discuss.grapheneos.org/d/12963-how-does-storage-scopes-work-dont-really-understand
- https://discuss.grapheneos.org/d/1935-how-do-i-enable-storage-scopes
- https://grapheneos.org/usage
- https://grapheneos.org/usage#storage-access
- https://grapheneos.org/faq
- https://grapheneos.org/usage#contact-scopes
- https://discuss.grapheneos.org/d/11361-just-discovered-contact-scopes
- https://discuss.grapheneos.org/d/27177-need-help-understanding-netguard

- https://discuss.grapheneos.org/d/16046-google-keyboard-w-network-permission-revoked
- https://discuss.grapheneos.org/d/22013-need-strong-clarification-regarding-network-permission-toggle
- https://discuss.grapheneos.org/d/27867-private-spaces-and-permissions
- https://grapheneos.org/releases
- https://grapheneos.org/
- https://www.reddit.com/r/GrapheneOS/comments/1pi2l5b/how_does_sandboxed_play_store_see_my_apps/
- https://discuss.grapheneos.org/d/22003-limiting-whatsapp-from-my-contacts
- https://discuss.grapheneos.org/d/11009-contact-scopes-whatsapp
- https://discuss.grapheneos.org/d/3731-step-by-step-guide-to-installing-whatsapp
- https://www.reddit.com/r/GrapheneOS/comments/1od2ws5/storage_scope_multimedia_permission_not_working/
- https://discuss.grapheneos.org/d/2914-storage-scope-not-working-for-whatsapp
- https://www.reddit.com/r/GrapheneOS/comments/163tbpi/gallery_app_picks_photos_from_music_library/
- https://discuss.grapheneos.org/d/29581-red-flags-permission-understanding-system-components-and-future-oem
- https://discuss.grapheneos.org/d/21783-need-help-from-someone-knowledgeable-with-setting-up-new-device-the-right-way

GrapheneOS ফরেনসিক ডিফেন্স রিসার্চ (GrapheneOS Forensic Defense Research)

- https://cellebrite.com/en/blog/the-solution-that-changed-modern-digital-investigations-forever/
- https://www.forensicscijournal.com/journals/jfsr/jfsr-aid1039.php
- https://law.stanford.edu/wp-content/uploads/2024/06/3.-Applegate-FINAL.pdf
- https://telefoncek.si/2024/05/2024-05-30-grapheneos-and-forensic-extraction-of-data/
- https://grapheneos.org/features
- https://osservatorionessuno.org/blog/2025/03/a-deep-dive-into-cellebrite-android-support-as-of-february-2025/

- https://discuss.grapheneos.org/d/20968-grapheneos-still-not-vulnerable-to-cellebrite-device-exploitation-as-of-feb-2025
- https://cellebrite.com/en/products/launches-releases/spring-release-2026/
- https://grapheneos.org/faq#encryption-keys-eviction
- https://blog.digital-forensics.it/2025/09/exploring-data-extraction-from-android.html
- https://lucidtruthtechnologies.com/bfu-vs-afu/
- https://discuss.grapheneos.org/d/24110-bfu-and-afu-password
- https://blogs.dsu.edu/digforce/2023/08/23/bfu-and-afu-lock-states/
- https://discuss.grapheneos.org/d/20883-a-recap-on-cellebrite-ufed-premium-and-graykey-capabilities-on-locked-devices
- https://discuss.grapheneos.org/d/19360-misconception-about-filesystem-based-encryption
- https://discuss.grapheneos.org/d/16758-help-me-understand-the-implications-of-bfu-and-afu
- https://osservatorionessuno.org/blog/2025/03/a-deep-dive-into-cellebrite-android-support-as-of-february-2025/#:~:text=After%20First%20Unlock%20(AFU)%3A,not%20been%20unlocked%20since%20boot.
- https://discuss.grapheneos.org/d/10108-pixel-afu-vs-bfu-security
- https://discuss.grapheneos.org/d/12848-claims-made-by-forensics-companies-their-capabilities-and-how-grapheneos-fares/71
- https://discuss.grapheneos.org/d/12848-claims-made-by-forensics-companies-their-capabilities-and-how-grapheneos-fares?page=5
- https://discuss.grapheneos.org/d/6787-general-info-for-unsupported-platforms-pure-diy-no-official-platform-support
- https://support.google.com/pixelphone/answer/7158570?hl=en
- https://grapheneos.org/faq
- https://www.reddit.com/r/GrapheneOS/comments/1o3fl5m/what_stops_attackers_from_brute_forcing_a_4_digit/
- https://discuss.grapheneos.org/d/13107-grapheneos-and-forensic-extraction-of-data
- https://www.reddit.com/r/GrapheneOS/comments/1q09jol/how_vulnerable_is_a_pixel_9a_running_graphene/#:~:text=In%20BFU%2C%20full%20encryption%20is,memory%2C%20

educing%20resistance%20to%20attacks.

- https://www.reddit.com/r/GrapheneOS/comments/1q09jol/how_vulnerable_is_a_pixel_9a_running_graphene/
- https://discuss.grapheneos.org/d/12848-claims-made-by-forensics-companies-their-capabilities-and-how-grapheneos-fares?page=7
- https://discuss.grapheneos.org/d/5140-afu-state
- https://www.sherlockforensics.com/blog/android-forensics-tool-comparison-2026.html
- https://discuss.privacyguides.net/t/claims-made-by-forensics-companies-their-capabilities-and-how-grapheneos-fares/18445
- https://securitylab.amnesty.org/latest/2025/02/cellebrite-zero-day-exploit-used-to-target-phone-of-serbian-student-activist/
- https://cellebrite.com/en/glossary/
- https://sharefoundation.info/en/spywares-first-step-a-systematic-analysis-of-exploits-used-for-mobile-device-compromise/
- https://madaidans-insecurities.github.io/guides/linux-hardening.html
- https://en.wikipedia.org/wiki/Cold_boot_attack
- https://blog.citp.princeton.edu/2008/02/21/new-research-result-cold-boot-attacks-disk-encryption/
- https://www.usenix.org/legacy/event/sec08/tech/full_papers/halderman/halderman.pdf?utm_source=Securitylab.ru
- https://discuss.grapheneos.org/d/22022-encrypted-ram-at-hardware-based-level
- https://discuss.grapheneos.org/d/17542-what-prevents-attackers-from-reading-out-ram-memory
- https://discuss.grapheneos.org/d/11860-vulnerabilities-exploited-in-the-wild-fixed-based-on-grapheneos-reports
- https://grapheneos.social/@GrapheneOS/112204443465440601
- https://discuss.grapheneos.org/d/10281-auto-reboot-question
- https://discuss.grapheneos.org/d/23736-automatic-18-hour-reboots
- https://debugging.works/blog/grapheneos-auto-reboot-feature-for-linux/
- https://discuss.grapheneos.org/d/12848-claims-made-by-forensics-companies-their-capabilities-and-how-grapheneos-fares/69

- https://discuss.grapheneos.org/d/24356-cold-brute-force-attack
- https://github.com/GrapheneOS/hardened_malloc
- https://synacktiv.com/en/publications/exploring-grapheneos-secure-allocator-hardened-malloc
- https://discuss.grapheneos.org/d/11196-hardened-malloc-functionality
- https://grapheneos.org/releases
- https://discuss.privacyguides.net/t/a-question-about-passwords-and-clipboard-in-linux/28170
- https://ventral.digital/posts/2024/12/9/practical-grapheneos-for-the-paranoid/
- https://grapheneos.social/@GrapheneOS/112826162113331497
- https://discuss.grapheneos.org/d/19816-usb-c-port-charging-only-how-much-security-does-it-actually-provide
- https://grapheneos.social/@GrapheneOS/112055235450847864
- https://discuss.grapheneos.org/d/10540-usb-peripheral-toggle-is-separate-from-the-standard-usb-gadget-mode
- https://grapheneos.org/features#usb-c-port-and-pogo-pins-control
- https://grapheneos.org/usage
- https://grapheneos.org/build
- https://discuss.grapheneos.org/d/21838-usb-c-off-feature-not-working-corrctly
- https://discuss.grapheneos.org/d/14755-take-out-usb-c-port-and-only-do-wireless-charging
- https://grapheneos.social/@GrapheneOS/111829820568926396
- https://discuss.privacyguides.net/t/updated-cellebrite-google-pixel-matrix-leak-february-2025/25911
- https://discuss.grapheneos.org/d/14344-cellebrite-premium-july-2024-documentation/168
- https://www.reddit.com/r/GrapheneOS/comments/1cv24pc/claims_made_by_forensics_companies_their/
- https://signal.org/blog/cellebrite-vulnerabilities/
- https://discuss.grapheneos.org/d/32745-safest-and-most-private-google-pixel-model
- https://sdpdtech.sandiego.gov/Cellebrite%20Inseyets%20(Previously%20Cellebrite%20Universal%20Extraction%20Device)/3+-+Surveillance+Impact+Report+-+Cellebrite+Universal+

Forensics+Extraction+Device+%28UFED%29.pdf

- https://forensic.manuals.mobiledit.com/MM/cellebrite-ufdr
- https://cao-94612.s3.us-west-2.amazonaws.com/documents/Cellebrite_UFED4PC_OverviewGuide_v7.66_July_2023.pdf
- https://www.scribd.com/document/698466425/Cellebrite-UFED-OverviewGuide
- https://grapheneos.org/releases#2026042100
- https://sharefoundation.info/wp-content/uploads/2025/09/Spyware.pdf
- https://www.forensicfocus.com/forums/general/cellebrite-error-s/
- https://www.reddit.com/r/computerforensics/comments/1bdw56n/cellebrite_error/
- https://www.ojp.gov/library/publications/cellebrite-ufed-version-1176-evaluation-report

GrapheneOS এক্সপ্লয়েট মিটিগেশন রিসার্চ (GrapheneOS Exploit Mitigation Research)

- https://en.wikipedia.org/wiki/Pegasus_(spyware)
- https://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/18763/?sequence=1&isAllowed=y
- https://discuss.grapheneos.org/d/19823-should-i-reboot-my-phone-daily-to-defend-against-zero-click-hackers
- https://www.reddit.com/r/GrapheneOS/comments/1nfbaxr/does_gos_protect_against_zero_click_text_message/
- https://nvd.nist.gov/vuln/detail/cve-2019-11932
- https://grapheneos.social/@GrapheneOS/113943816103474971
- https://www.reddit.com/r/GrapheneOS/comments/16ikhlj/does_graphene_protect_against_0_day/
- https://citizenlab.ca/research/forcedentry-nso-group-imessage-zero-click-exploit-captured-in-the-wild/
- https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-technical-analysis.pdf
- https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus/
- https://www.trendmicro.com/en_us/research/21/i/analyzing-pegasus-spywares-zero-click-iphone-exploit-forcedentry.html

- https://www.securityweek.com/whatsapp-flaw-allows-remote-code-execution-malicious-gif-file/
- https://wizardcyber.com/whatsapp-android-zero-click-media-exploit/
- https://www.trendmicro.com/en_us/research/19/k/patched-gif-processing-vulnerability-cve-2019-11932-still-afflicts-multiple-mobile-apps.html
- https://security.paloaltonetworks.com/CVE-2023-4863
- https://nvd.nist.gov/vuln/detail/cve-2023-4863
- https://www.wiz.io/blog/cve-2023-4863-and-cve-2023-5217-exploited-in-the-wild
- https://blog.cloudflare.com/uncovering-the-hidden-webp-vulnerability-cve-2023-4863/
- https://discuss.grapheneos.org/d/11196-hardened-malloc-functionality
- https://synacktiv.com/en/publications/exploring-grapheneos-secure-allocator-hardened-malloc
- https://www.cape.co/blog/grapheneos-privacy-and-security-features
- https://github.com/GrapheneOS/hardened_malloc
- https://discuss.grapheneos.org/d/9289-pegasus-and-predator-spyware/5
- https://grapheneos.org/releases
- https://discuss.grapheneos.org/d/8439-mte-support-status-for-grapheneos?page=4
- https://news.ycombinator.com/item?id=38125887
- https://discuss.grapheneos.org/d/19060-information-on-cfi-in-grapheneos-and-response-to-a-paper
- https://grapheneos.org/features
- https://www.reddit.com/r/GrapheneOS/comments/1sdcd5d/psa_is_using_grapheneos_like_stock_android_okay/
- https://discuss.grapheneos.org/d/7615-understanding-grapheneos-sandbox-and-permissions-in-vulnerability-scenarios
- https://discuss.grapheneos.org/d/29453-help-me-understand-this-google-play-services-being-sandboxed
- https://discuss.grapheneos.org/d/12963-how-does-storage-scopes-work-dont-really-understand
- https://grapheneos.org/usage
- https://en.wikipedia.org/wiki/Drive-by_download

- https://arcticwolf.com/resources/blog/understanding-drive-by-download-attacks/
- https://discuss.grapheneos.org/d/28173-cant-disable-webassembly-on-vanadium
- https://www.reddit.com/r/GrapheneOS/comments/1qmvaf4/vanadium_most_privatesecure/
- https://grapheneos.social/@GrapheneOS/112708050929472969
- https://discuss.grapheneos.org/d/14334-vanadium-update-cant-login-to-some-sites
- https://zerotrace.org/kb/features/
- https://discuss.grapheneos.org/d/9700-vanadium-site-isolation-sandboxing
- https://www.reddit.com/r/GrapheneOS/comments/13t65l0/isolating_sites_on_vanadium_should_i_use_separate/
- https://www.usenix.org/conference/usenixsecurity19/presentation/reis
- https://sharefoundation.info/wp-content/uploads/2025/09/Spyware.pdf
- https://news.ycombinator.com/item?id=32003756
- https://grapheneos.org/faq

GrapheneOS প্রজেক্ট রিসার্চ ব্রিফ (GrapheneOS Project Research Brief)

- https://grapheneos.org/
- https://grapheneos.org/features
- https://www.youtube.com/watch?v=Y5GrbhB2HHQ
- https://en.wikipedia.org/wiki/GrapheneOS
- https://grapheneos.org/faq
- https://discuss.grapheneos.org/d/32656-motorola-partnership-announcement/62
- https://motorolanews.com/motorola-three-new-b2b-solutions-at-mwc-2026/
- https://www.privacyfirst.com.au/history-of-grapheneos/
- https://grapheneos.org/history/legacy-changelog
- https://discuss.grapheneos.org/d/34369-original-grapheneos-responses-to-wired-fact-checker
- https://en.wikipedia.org/wiki/CopperheadOS
- https://www.reddit.com/r/privacy/comments/klbjhu/just_a_reminder_that_grapheneos_is_being_sued_by/

- https://grapheneos.org/history/
- https://news.ycombinator.com/item?id=36089104
- https://discuss.privacyguides.net/t/daniel-micay-publicly-steps-down-as-project-leader-of-grapheneos/12677
- https://www.reddit.com/r/thehatedone/comments/13uyd0f/daniel_micay_steps_down_as_the_leader_of/
- https://news.ycombinator.com/item?id=47215973
- https://discuss.grapheneos.org/d/32656-motorola-partnership-announcement
- https://piunikaweb.com/2026/04/22/grapheneos-blasts-wired-over-daniel-micay-james-donaldson-feud-story/
- https://discuss.grapheneos.org/d/5864-grapheneos-leadership
- https://discuss.privacyguides.net/t/grapheneos-will-drop-google-pixel-exclusivity-with-major-snapdragon-powered-devices-coming/32097
- https://discuss.grapheneos.org/u/tango/likes
- https://music.youtube.com/podcast/Y5GrbhB2HHQ
- https://en.xiaomitoday.it/categories/news/Italian-and-international-brands/motorola
- https://www.reddit.com/r/GooglePixel/comments/1421l92/tensor_g3_google_pixel_8_series_chipset_fails_to/
- https://www.reddit.com/r/degoogle/comments/1rilwdv/grapheneos_partnership_with_motorola_lenovo_now/

GrapheneOS টেকনিক্যাল অ্যাপেন্ডিক্স রিসার্চ (GrapheneOS Technical Appendix Research)

- https://en.wikipedia.org/wiki/GrapheneOS
- https://grapheneos.org/faq
- https://grapheneos.org/build
- https://grapheneos.org/install/cli
- https://github.com/MuratovAS/grapheneos-magisk
- https://grapheneos.org/install/web

- https://discuss.grapheneos.org/d/21104-what-specs-do-the-grapheneos-devs-use-for-their-local-development
- https://grapheneos.org/features
- https://grapheneos.org/install/
- https://www.reddit.com/r/GrapheneOS/comments/1ohijri/what_should_one_choose_if_grapheneos_stops/
- https://grapheneos.org/articles/server-traffic-shaping
- https://www.bufferbloat.net/projects/codel/wiki/Cake/
- https://www.shieldmdm.io/post/shieldmdm-independent-device-owner-mode-for-android-grapheneos-and-ios
- https://goodancestor.com/guide-to-setting-up-graphene-os/
- https://joelgsamuel.medium.com/risky-smartphone-travel-997ef349ccde
- https://discuss.grapheneos.org/d/4411-does-gos-play-well-with-mdm-for-corporateenterprise-environments
- https://grapheneos.org/usage
- https://discuss.grapheneos.org/u/headwind-mdm
- https://discuss.grapheneos.org/d/13587-mdm-on-grapheneos
- https://qa.h-mdm.com/14300/how-install-headwind-grapheneos-signing-agent-platform-keys
- https://android.googlesource.com/platform/frameworks/base/+/master/core/java/android/app/admin/DevicePolicyManager.java
- https://developer.android.com/work/device-management-policy
- https://discuss.grapheneos.org/d/22953-graphene-os-edition-for-high-risk-people
- https://discuss.grapheneos.org/d/28682-maximum-security-and-anonymity
- https://www.reddit.com/r/GrapheneOS/comments/1ra2noj/is_this_the_most_private_way_to_configure/
- https://discuss.grapheneos.org/d/2185-grapheneoss-recommendation-on-dns-servers
- https://www.youtube.com/watch?v=eyWmcItzisk
- https://developer.android.com/identity/data/autobackup
- https://calyxinstitute.org/projects/seedvault-encrypted-backup-for-android
- https://www.reddit.com/r/GrapheneOS/comments/13up3gt/full_backup_of_phone/

- https://discuss.grapheneos.org/d/22631-how-could-i-backup-my-graphenos-users
- https://discuss.grapheneos.org/d/32630-my-backup-restore-thread-tips-tricks-my-experience
- https://github.com/seedvault-app/seedvault/discussions/406
- https://discuss.grapheneos.org/d/8291-restoring-seedvault-backup-from-pixel-6-pro-on-a-new-pixel-8-pro
- https://www.reddit.com/r/GrapheneOS/comments/1rg0yrk/seedvault_backup_to_selfhosted_webdav_docker/
- https://medium.com/@glenn.sorrentino/host-your-own-mobile-backups-with-seedvault-and-nextcloud-dd3bb1632487
- https://discuss.grapheneos.org/d/9804-where-are-backups-stored
- https://www.reddit.com/r/Calyx/comments/ilddx7/seedvault_backup_location/
- https://discuss.grapheneos.org/d/32378-what-is-simplest-backup-option-for-my-use-case
- https://community.iode.tech/t/how-is-seedvault-supposed-to-work/7751
- https://grapheneos.org/articles/attestation-compatibility-guide
- https://discuss.grapheneos.org/d/11799-remote-attestation-mechanics
- https://attestation.app/tutorial
- https://github.com/GrapheneOS/AttestationServer
- https://www.reddit.com/r/GrapheneOS/comments/1430ni2/auditor_remote_attestation_for_everyone/

উপসংহার: প্রাইভেসি বা গোপনীয়তার ভবিষ্যৎ (Conclusion: The Future of Privacy)

যখন আমরা এই বইটি শুরু করেছিলাম, তখন আপনি আক্ষরিক অর্থেই একটি কাচের ঘরের (glass house) ভেতর দাঁড়িয়ে ছিলেন।

আপনার স্মার্টফোন—সিলিকন এবং কাচের তৈরি সেই সুদৃশ্য স্লেটটি, যা আপনি দিনে ষোল ঘণ্টা নিজের শরীরের সাথে নিয়ে ঘোরেন—সেটি মূলত ছিল একটি সার্ভিল্যান্স ডিভাইস (surveillance device)। এটি প্রতি চার মিনিট অন্তর আপনার ফিজিক্যাল কোঅর্ডিনেটস বা ভৌত অবস্থান লগ করত। এটি আপনার ব্যক্তিগত কথোপকথনের বিষয়বস্তু ইনডেক্স করত। এটি আপনার তোলা ফটোগ্রাফগুলোর চেহারাগুলো ক্যাটালগ করত, আপনার সামাজিক সম্পর্কের টপোলজি ম্যাপ করত এবং আপনার আচরণগত টেলিমেট্রির (behavioral telemetry) একটি নিরবচ্ছিন্ন স্রোত পৃথিবীর অন্য প্রান্তে থাকা কর্পোরেট সার্ভারগুলোতে ট্রান্সমিট বা প্রেরণ করত।

এই আর্কিটেকচার বা কাঠামোর জন্য আপনি কোনো সম্মতি বা কনসেন্ট (consent) দেননি। আপনি বক্সটি খোলার আগেই এটি ডিজাইন করা হয়েছিল। এর অপারেটিং সিস্টেম, প্রি-ইনস্টলড অ্যাপগুলো এবং ব্যাকগ্রাউন্ড সার্ভিসগুলো সিলিকন লেভেল থেকে এমনভাবে তৈরি করা হয়েছিল, যার একমাত্র অর্থনৈতিক উদ্দেশ্য (economic imperative) ছিল: আপনার অস্তিত্বের এক্সট্র্যাকশন (extraction), অ্যাগ্রিগেশন (aggregation) এবং মনেটাইজেশন (monetization)। আপনি কোনো কাস্টমার বা গ্রাহক ছিলেন না। বরং আপনি নিজেই ছিলেন প্রোডাক্ট (product)।

আমরা এই আর্কিটেকচারটিকে বলেছিলাম ডিজিটাল প্যানোপটিকন (Digital Panopticon)—এমন এক সর্বগ্রাসী, পরিবেষ্টিত এবং স্বাভাবিককৃত সার্ভিল্যান্স সিস্টেম, যার ভেতরে বিশ্বের সিংহভাগ মানুষ বাস করলেও তারা টেরই পায় না যে তারা নজরদারির ভেতরে আছে।

কিন্তু আপনি তা বুঝতে পেরেছিলেন। এবং আপনি সেখান থেকে বেরিয়ে আসার সিদ্ধান্ত নিয়েছিলেন।

দ্য জার্নি (The Journey)

অত্যন্ত নিখুঁতভাবে আর্কিটেক্ট বা তৈরি করা তিনটি পর্ব বা সেকশনের মধ্য দিয়ে, আপনি আপনার ডিজিটাল অস্তিত্বের এক সম্পূর্ণ ট্রান্সফরমেশন বা রূপান্তর ঘটিয়েছেন।

পর্ব ১: দ্য ফাউন্ডেশন (Part I: The Foundation)-এ, আপনি আপনার হার্ডওয়্যারের ওপর নিজের নিয়ন্ত্রণ পুনরুদ্ধার করেছেন। আপনি জেনেছেন কেন GrapheneOS Pixel প্ল্যাটফর্ম বেছে নিয়েছে—গুগলের প্রতি কোনো আনুগত্য থেকে নয়, বরং Titan M2 secure element, আইসোলেটেড বেসব্যান্ড এবং Verified Boot chain-এর কারণে, যা একটি সত্যিকারের হার্ডেনড (hardened) OS সাপোর্ট করার জন্য একমাত্র সক্ষম সিলিকন। আপনি ফার্মওয়্যার ফ্ল্যাশ করেছেন, bootloader লক করেছেন এবং আপনার নিজস্ব cryptographic keys এনরোল করেছেন। আপনি Auditor app দিয়ে ইনস্টলেশনটি ভেরিফাই করেছেন এবং অপারেশনাল ডিসিপ্লিনের (operational discipline) দিকে একটি ক্রিটিক্যাল শিফট বা অত্যন্ত গুরুত্বপূর্ণ পরিবর্তন শুরু করেছেন।

পর্ব ২: দ্য ফোর্ট্রেস (Part II: The Fortress)-এ, আপনি আপনার অভ্যন্তরীণ সুরক্ষাব্যবস্থা বা ডিফেন্স আর্কিটেক্ট করেছেন। আপনি Google Play Services-কে স্যান্ডবক্স (sandboxed) করেছেন—যার ফলে এটি প্রায় নিঁখুত অ্যাপ্লিকেশন কম্প্যাটিবিলিটি বজায় রাখলেও সিস্টেমের সমস্ত প্রিভিলেজ (privilege) থেকে বঞ্চিত হয়েছে। আপনি হোস্টাইল (hostile) অ্যাপ্লিকেশনগুলোকে একটি বানোয়াট রিয়েলিটি (fabricated reality) ফিড করার জন্য Storage Scopes এবং Contact Scopes ডিপ্লয় করেছেন। আপনি পার-অ্যাপ কিল সুইচ (per-app kill switches) দিয়ে আনঅথরাইজড নেটওয়ার্ক কানেকশন ছিন্ন করেছেন, আপনার DNS কোয়েরিগুলো এনক্রিপ্ট করেছেন এবং হার্ডেনড VPN tunnels-এর ভেতর দিয়ে আপনার ট্রাফিক রাউট (route) করেছেন।

আপনি আপনার ডিজিটাল জীবনকে সম্পূর্ণ আইসোলেটেড বা বিচ্ছিন্ন User Profiles-এ কম্পার্টমেন্টালাইজ (compartmentalize) করেছেন—যেখানে প্রতিটির রয়েছে নিজস্ব কি (keys), নিজস্ব অ্যাপ এবং নিজস্ব ডেস্ট্রাকশন প্রোটোকল (destruction protocol)। আপনি Deadman's Switch, Duress PIN এবং USB-C পোর্ট লকডাউন কনফিগার করেছেন। আপনি ফিশিং-প্রুফ (phishing-proof) FIDO2 হার্ডওয়্যার কি দিয়ে আপনার অথেনটিকেশন সুরক্ষিত করেছেন।

পর্ব ৩: দ্য ব্যাটলফিল্ড (Part III: The Battlefield)-এ, আপনি এই সুরক্ষাব্যবস্থাগুলোকে ফায়ারের (fire) বা আক্রমণের মুখে পরীক্ষা হতে দেখেছেন। আপনি দেখেছেন কীভাবে একটি Cellebrite টার্মিনাল একটি বিচ্ছিন্ন USB পোর্টের মাধ্যমে ডিভাইসটি ডিটেক্ট করতে ব্যর্থ হয়। আপনি পর্যবেক্ষণ করেছেন কীভাবে একটি Pegasus-class zero-click exploit সজোরে আছড়ে পড়ে ভেঙে যায় `hardened_malloc`-এর গার্ড পেজ এবং ARM MTE-এর হার্ডওয়্যার-এনফোর্সড মেমরি ট্যাগের সামনে। আপনি দেখেছেন কীভাবে একটি Stingray-এর ডাউনগ্রেড অ্যাটাক (downgrade attack) LTE-Only মোড দ্বারা প্রত্যাখ্যাত হয় এবং কীভাবে Titan M2 সিলিকনে ফিউজ করা ইমিউটেবল (immutable) Verified Boot chain একটি সাপ্লাই চেইন অ্যাটাক (supply chain attack) প্রত্যাখ্যান করে।

প্রতিটি অ্যাটাক নিউট্রালাইজ (neutralized) বা নিষ্ক্রিয় করা হয়েছে। প্রতিটি কিল চেইন (kill chain) ভেঙে দেওয়া হয়েছে। ভাগ্যের জোরে নয়—বরং গণিতের জোরে।

দ্য ফিউচার: পিক্সেলের সীমানা পেরিয়ে (The Future: Beyond Pixel)

এক দশকেরও বেশি সময় ধরে GrapheneOS একটিমাত্র হার্ডওয়্যার ফ্যামিলির সাথে সমার্থক ছিল: Google Pixel। এই এক্সক্লুসিভিটি (exclusivity) বা একচেটিয়া অধিকার কোনো পছন্দ থেকে জন্ম নেয়নি, বরং এটি ছিল একটি প্রয়োজন (necessity)। প্রজেক্টের আপসহীন হার্ডওয়্যার সিকিউরিটি রিকোয়ারমেন্টস পূরণ করতে পারে এমন একমাত্র কমার্শিয়ালি অ্যাভেইলেবল বা বাণিজ্যিকভাবে উপলব্ধ প্ল্যাটফর্ম ছিল পিক্সেল লাইনআপ (Pixel lineup)।

সেই যুগের অবসান হতে চলেছে।

২০২৫ সালে ঘোষিত GrapheneOS এবং **Motorola Mobility**-এর মধ্যকার ল্যান্ডমার্ক পার্টনারশিপটি (landmark partnership) এই প্রজেক্টের ইতিহাসের সবচেয়ে তাৎপর্যপূর্ণ বিস্তৃতি। GrapheneOS-এর কঠোর হার্ডওয়্যার স্পেসিফিকেশনগুলো পূরণ করে এমন ডিভাইস তৈরি করতে মটোরোলা (Motorola) প্রতিশ্রুতিবদ্ধ হয়েছে: সঠিক IOMMU এনফোর্সমেন্ট, একটি হার্ডওয়্যার-ব্যাকড সিকিউর এলিমেন্ট, কাস্টম AVB কি এনরোলমেন্ট এবং এমন একটি bootloader যা থার্ড-পার্টি কি দ্বারা লক করা সম্ভব।

এই পার্টনারশিপ পিক্সেল এক্সক্লুসিভিটির বাধা ভেঙে দিয়েছে। এটি হার্ডেনড মোবাইল হার্ডওয়্যার মার্কেটে প্রতিযোগিতার জন্ম দিয়েছে। এটি উন্নয়নশীল দেশগুলোর সাংবাদিক, স্বৈরাচারী শাসনাধীনে থাকা অ্যাক্টিভিস্ট এবং এন্টারপ্রাইজ সিকিউরিটি টিমগুলোকে অনেক সাশ্রয়ী মূল্যে এবং আরও ব্যাপকভাবে ডিস্ট্রিবিউট হওয়া হার্ডওয়্যারে GrapheneOS অ্যাক্সেস করার সুযোগ করে দিয়েছে। ফোর্ট্রেস বা দুর্গটি এখন আর কোনো সিঙ্গেল ম্যানুফ্যাকচারারের (manufacturer) ইকোসিস্টেমে সীমাবদ্ধ নেই।

এর ইমপ্লিকেশন বা প্রভাব অত্যন্ত গভীর। যত বেশি হার্ডওয়্যার ম্যানুফ্যাকচারার ভেরিফাইএবলি সিকিউর (verifiably secure) ডিভাইসের প্রতি এই মার্কেট ডিমান্ডটি (market demand) উপলব্ধি করবে, প্রবেশের খরচ বা কস্ট অব এন্ট্রি (cost of entry) ততটাই কমে আসবে। GrapheneOS তখন আর শুধুমাত্র এলিট অপারেটরদের জন্য কোনো নিস টুল (niche tool) থাকবে না, বরং তা প্রাইভেসি বা গোপনীয়তার মৌলিক অধিকারের মূল্যায়ন করে এমন যেকোনো মানুষের জন্য একটি অ্যাক্সেসিবল (accessible) এবং মেইনস্ট্রিম অপশনে পরিণত হবে।

দ্য রাইট টু প্রাইভেসি (The Right to Privacy)

১৯৪৮ সালে, জাতিসংঘ (United Nations) সর্বজনীন মানবাধিকার ঘোষণাপত্র (Universal Declaration of Human Rights) গ্রহণ করে। এর অনুচ্ছেদ ১২ (Article 12)-এ বলা হয়েছে:

> *"কারো গোপনীয়তা, পরিবার, বসতবাড়ি বা চিঠিপত্রের ওপর খেয়ালখুশিমতো হস্তক্ষেপ করা যাবে না।" ("No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence.")*

এই কথাগুলো লেখার আটাত্তর বছর পর, আধুনিক বিশ্বের প্রযুক্তিগত অবকাঠামোকে সিস্টেমেটিক্যালি বা নিয়মতান্ত্রিকভাবে এমনভাবে তৈরি করা হয়েছে, যাতে এই অধিকারটিকে কল্পনাতীত স্কেলে লঙ্ঘন করা যায়। প্রতিটি আনএনক্রিপ্টেড টেক্সট মেসেজ, নীরবে হারভেস্ট (harvest) হওয়া প্রতিটি GPS কোঅর্ডিনেট এবং চুরি করা কন্ট্যাক্ট লিস্ট (contact list) দিয়ে তৈরি করা প্রতিটি শ্যাডো প্রোফাইল (shadow profile) হলো Article 12-এর একটি লঙ্ঘন। এটি কোনো তাত্ত্বিক লঙ্ঘন নয়। এটি একটি গাণিতিক, পরিমাপযোগ্য এবং ডকুমেন্টেড বা দলিলায়িত লঙ্ঘন—যা পৃথিবীর প্রতিটি মহাদেশে, কোটি কোটি ডিভাইসে, প্রতিদিন বিলিয়ন বিলিয়ন বার ঘটছে।

GrapheneOS হলো এই মানবাধিকারটির একটি টেকনোলজিকাল এনফোর্সমেন্ট (technological enforcement)।

এটি কোনো প্রোডাক্ট বা পণ্য নয়। এটি কোনো ব্র্যান্ডও নয়। এটি হলো একটি আর্কিটেকচারাল প্রুফ বা প্রমাণ যে মানুষ এবং তার ডিভাইসের মধ্যে সম্পূর্ণ ভিন্ন একটি সম্পর্ক গাণিতিকভাবে সম্ভব। এমন একটি সম্পর্ক যেখানে হার্ডওয়্যার তার অপারেটরকে সার্ভ বা সেবা করবে, কোনো ডেটা ব্রোকারকে নয়। যেখানে এনক্রিপশন কি-গুলোর মালিক হবে সাধারণ ব্যক্তি, কোনো কর্পোরেশন নয়। যেখানে অপারেটিং সিস্টেমের প্রধান কাজ হবে ব্যবহারকারীর সার্বভৌমত্ব (sovereignty) রক্ষা করা, তার আচরণগত উদ্বৃত্ত (behavioral surplus) এক্সট্র্যাক্ট করা নয়।

সার্ভিল্যান্স ইকোনমি (surveillance economy) গত দুই দশক ধরে ইতিহাসের সবচেয়ে বিস্তৃত মানব মনিটরিং অ্যাপারেটাস (monitoring apparatus) বা নজরদারি ব্যবস্থা তৈরি করেছে। তারা বিলিয়ন বিলিয়ন ডলার খরচ করেছে। তারা হাজার হাজার ইঞ্জিনিয়ার নিয়োগ দিয়েছে। তারা বিশ্বের চার বিলিয়ন মানুষের পকেটে তাদের কোড এম্বেড (embed) করেছে।

আর একটি ছোট, তীব্রভাবে স্বাধীন ওপেন-সোর্স প্রজেক্ট—যা অনুদানের টাকায় চলে, নীতির ওপর ভর করে এগিয়ে যায় এবং গণিত ছাড়া যার আর কোনো অস্ত্র নেই—এমন একটি সিস্টেম তৈরি করেছে যা তাদের সম্পূর্ণ অ্যাপারেটাসকে অকেজো বা অপ্রাসঙ্গিক (irrelevant) করে দিয়েছে।

সেই সিস্টেমটিই আজ আপনি আপনার হাতে ধরে আছেন।

কাচের ঘরটি এখন আপনার পেছনে। দুর্গটি এখন আপনার। প্যানোপটিকন-এর অন্য পাশে আপনাকে স্বাগতম (Welcome to the other side of the panopticon)।

লেখক পরিচিতি (About the Author)

রকিবুল ইসলাম রনি (Rokibul Islam Roni) হলেন একজন পেনিট্রেশন টেস্টার (penetration tester) যিনি অফেনসিভ সিকিউরিটি (offensive security) এবং রেড টিম অপারেশনে (red team operations) বিশেষজ্ঞ।

- **ওয়েবসাইট:** rokibulroni.com
- **ইমেইল:** mail@rokibulroni.com
- **সর্বশেষ রিলিজ চেক করুন:** https://gos.rokibulroni.com/books/